REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE.

Ministre de l'enseignement supérieur et de la recherche scientifique.

Université Mouloud Mammeri Tizi-ouzou.

Faculté du génie électrique et informatique.

Département informatique.

Mémoire de Master II en informatique

Option: Conduite de projet informatique

Thème:

Génération de structures dans les réseaux sociaux

Développé par : **Boudiba Tahar-Rafik**

Dirigé par : R.Ahmed Ouamer

Membres du jury:

- M[r] M. Daoui
- M[r] A. Hammache
- M[r] M. Ramdane

« Promotion : 2010-2011 »

Abstract:

The search for information in the social Web must consider at the same time the structure and the contents of the social interactions between the users. Languages like RDF or OWL are used for the representation of the contents.

The aim of this work is to propose techniques exploiting the contents to establish connections between network's communities and activities of various communities. It is question of considering the treatment of information in collaborative environments, like blogs, virtual offices, centers of interests or an e-learning plate form.

The final objective of this project is to conceive and develop techniques making it possible to put in correspondences communities, individuals, activities, etc in a social network.

Keywords: Social network - Ontology - Semantic web -e-learning - collective training.

Résumé :

La recherche d'information dans le Web social doit considérer à la fois la structure et le contenu des interactions sociales entre les utilisateurs. Des langages comme RDF ou OWL sont utilisés pour la représentation du contenu.

Dans ce travail, il s'agit de proposer des techniques exploitant le contenu pour établir des connections entre communautés dans le réseau et entre activités de différentes communautés. Il s'agit de considérer le traitement d'informations dans les environnements collaboratifs, comme les blogs, les bureaux virtuels, les centres d'intérêts ou encore les plates formes d'e-learning.

L'objectif final de ce travail est de concevoir et de développer des techniques permettant de mettre en correspondances des communautés, individus, activités, etc. dans un réseau social.

Mots clés : Réseaux sociaux – Ontologie – Web Sémantique - e-learning - Apprentissage collectif.

Liste des Figures :

Liste des tableaux :

Sommaire :

Chapitre I : Le web sémantique

Chapitre II : Les réseaux sociaux

Chapitre III : Conception

Chapitre IV : Réalisation

Introduction générale :

Au-delà de ses acquis déjà nombreux l'un des principaux objectifs du Web sémantique est de permettre aux utilisateurs d'utiliser la totalité du potentiel du Web : ainsi, ils pourront trouver, partager et combiner des informations plus facilement.

Aujourd'hui tout le monde est capable d'utiliser des forums, d'utiliser des réseaux sociaux, de faire des recherches ou même d'acheter différents produits.

Néanmoins, il serait mieux que la machine fasse tout ceci à la place de l'homme, car actuellement, les machines ont besoins de l'homme pour effectuer ces tâches. La raison principale est que les pages Web actuelles sont conçues pour être lisible par des êtres humains et non par des machines.

Le Web sémantique a donc comme principal objectif que ces mêmes machines puissent réaliser seules toutes les tâches fastidieuses comme la recherche ou l'association d'informations et d'agir sur le Web lui-même.

Le présent mémoire vise à :

- ✓ Offrir un environnement communautaire permettant le partage de ressources.
- ✓ Gérer ce réseau social et donc assurer : La gestion des membres du réseau.

Et enfin développer une approche de recherche basée sur le profil utilisateur dont ces centres d'intérêts sont représentés par une ontologie de domaine.

Ce mémoire comporte quatre chapitres :

- Le premier chapitre présente un état d'art sur le web sémantique ainsi que ses diverses applications, composants et fonctionnalités.
- Le second est consacré aux réseaux sociaux sur et les interactions des utilisateurs au travers des usages du web s'étalant ainsi sur la relation du web social et du web sémantique.
- Le troisième chapitre présente la conception du réseau social et l'ontologie conçue dans le cadre de ce projet.
- Enfin le quatrième chapitre décrit l'environnement de développement et l'implémentation de notre réseau.
- Deux annexes seront ajoutés ; Annexe A et annexe B décrivant respectivement Le langage de modélisation UML et de structuration XML.

Chapitre I : Le web sémantique

I. Introduction [1]

A travers Internet, un très grand nombre de services et de documents est accessible à tous les usagers. La plupart des services et documents fournis actuellement proposent une organisation, un contenu, un mode d'interaction et une présentation uniques pour tous les utilisateurs. Ceci peut être suffisant dans certains cas. Cependant les utilisateurs ne sont pas intéressés tous par les mêmes informations et n'ont pas les mêmes attentes, connaissances, compétences, centres d'intérêts, etc. Ils ne saisissent pas toujours que des services et des documents ne sont pas tous conçus à l'origine à leur intention. La raison à cela est que l'organisation, le contenu, les modes d'interaction de ces documents et services sont inadaptés à leurs besoins.

Parmi les problèmes qui se posent on peut citer :

- l'accès à l'information pertinente.
- la navigation dans un grand espace de ressources.
- la compréhension d'une ressource complexe (site web, services).

II. Evolution du web [2]

Le Web est essentiellement un espace volumineux d'information accessible par le réseau Internet. Se construisant sur le protocole HTTP à ses origines qui permet d'échanger des ressources accessibles en ligne : documents, images, fichiers multimédias…

Le format le plus utilisé sur le web est HTML. Il permet de représenter des documents hypertextes avec des balises de marquage et ainsi des hyperliens entre eux.

Le web ne se limite pas au protocole HTTP. Il peut s'étendre avec d'autres protocoles : FTP, SMTP. De même HTML, n'est pas le seul format des documents du Web.XML qui est un langage standardisé pour la représentation des données et documents structurés sur le Web. N'ayant pas de sémantique formelle il peut s'appliquer à n'importe quelle application.

Les acquis du Web sont déjà nombreux, et pourrait évoluer sur deux axes (selon son degrés de connectivité d'information et celui de sa connectivité social).

- Le premier axe d'évolution majeure et celle du Web sémantique.
- Le deuxième a pour but de devenir un moyen facile et rapide de développer les relations sociales.

III. Le web sémantique

L'expression Web sémantique, fait d'abord référence à la vision du Web de demain comme un vaste espace d'échange de ressources entre êtres humains et machines permettant une exploitation, qualitativement supérieure, de grands volumes d'informations et de services variés.

Il devrait voir, à la différence du Web que nous connaissons aujourd'hui, les utilisateurs déchargés d'une bonne partie de leurs tâches de recherche, de construction et de combinaison des résultats.

Le Web actuel est essentiellement syntaxique, dans le sens que la structure des documents (ou ressources au sens large) est bien définie, mais que son contenu reste quasi inaccessible aux traitements machines. Seuls les humains peuvent interpréter leurs contenus. La nouvelle génération de Web – Le Web sémantique – a pour ambition de lever cette difficulté. Les ressources du Web seront plus aisément accessibles aussi bien par l'homme que par la machine, grâce à la représentation sémantique de leurs contenus.

Ainsi le Web sémantique répond aux problèmes soulignés précédemment à savoir :

- La recherche d'information sur Internet peut grandement être améliorée et peut mieux cibler les résultats en fonction des besoins utilisateurs.
- Guider l'utilisateur dans ce grand espace d'information : adapter l'accès à l'information en fonction de ses besoins.

- Augmenter la cohérence des documents et services proposées dans le web, et fournir à l'utilisateur des repères l'aidant à identifier les composants majeurs du document ou du service et sa structure globale.

Le Web sémantique, concrètement, est une infrastructure permettant l'utilisation de connaissances formalisées en plus du contenu informel actuel du Web, même si aucun consensus n'existe sur jusqu'où cette formalisation doit aller.

Cette infrastructure doit permettre d'abord de localiser, d'identifier et de transformer des ressources de manière robuste et saine tout en renforçant l'esprit d'ouverture du Web avec sa diversité d'utilisateurs.

Elle doit :

- S'appuyer sur un certain niveau de consensus portant, par exemple, sur les langages de représentation ou sur les ontologies utilisées.

- Contribuer à assurer, le plus automatiquement possible, l'interopérabilité et les transformations entre les différents formalismes et les différentes ontologies.

- Faciliter la mise en œuvre de calculs et de raisonnements complexes tout en offrant des garanties supérieures sur leur validité. Elle doit offrir des mécanismes de protection (droits d'accès, d'utilisation et de reproduction), ainsi que des mécanismes permettant de qualifier les connaissances afin d'augmenter le niveau de confiance des utilisateurs.

IV. Architecture du Web sémantique [2]

L’architecture du Web sémantique s’appuie sur des langages représentant des connaissances sur le web. S'articulant sur une architecture en couches celui-ci permet une approche graduelle dans les processus de standardisation et d’acceptation par les utilisateurs.

Un langage de la couche haute doit être une extension du langage de la couche au dessous.

Ci-dessous une liste introduisant la fonction principale de chaque couche dans l’architecture du Web sémantique :

- XML : est utilisé comme couche de base syntaxique du web sémantique. Le langage XML est actuellement considéré comme un standard de transport de données sur le web.
- La couche RDF représente les métadonnées pour les ressources web.
- La couche « ontologie », fondé sur une formalisation commune, spécifie la sémantique de métadonnées fournies dans le web sémantique.
- La couche « logique » s’appuie sur des règles d’inférences qui permettent le raisonnement intelligent exécuté par des agents logiciels.
- Les couches « preuve » et « confiance » supportent un mécanisme de communication inter-agents pour valider les résultats de raisonnement. Cela pourra assurer la fiabilité des services automatiques du Web sémantique.

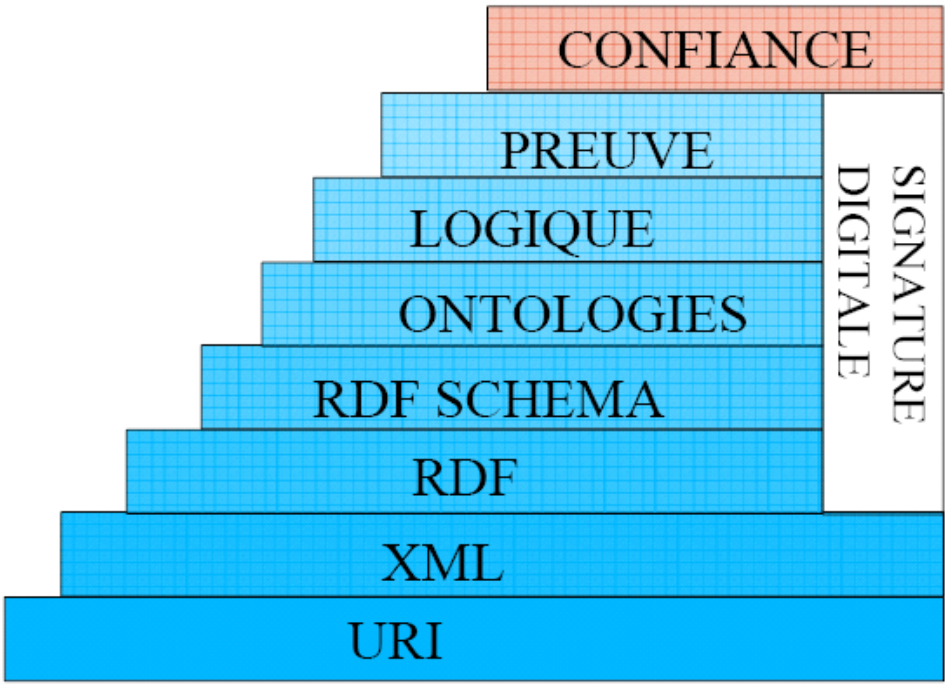


Figure I.1. Les couches du Web sémantique [3].

IV.1. Couche de base (URI)

Un aspect central du Web est sa capacité d'identification et de localisation des diverses ressources.

Elle repose sur la notion d'URI (Uniform Resource Identifier) qui permet d'attribuer un identificateur unique à un ensemble de ressources (virtuelles ou adressables).Il en existe deux spécialisations :

- Les URL (Uniform Ressource Locations) qui supposent un système d'adressage universel.

- Les URN (Uniform Ressource Names) qui supposent un système de nommage universel.

Les ressources accessibles sur le web sont encore le plus souvent adressées par des URL.

IV.2. XML

eXtensible Mark up Langages est un langage de balise permettant de structurer des données et/ou des documents sur le Web .Le langage XML est interopérable du point de vue syntaxique. Il sert à représenter des données échangeables sur le web.

Ainsi, tous les langages du web sémantique sont systématiquement exprimables et échangeables dans une syntaxe XML. En bref, XML permet aux utilisateurs d'ajouter une structure arbitraire à leurs documents sans rien dire de la signification – de la sémantique- de ces structures **[4]**.

IV.2.1. Le principe de l'XML

Sa structure est arborescente et très similaire au principe de l'orienté objet, chaque type de document (article scientifique, bon de commande, document administratif, graphique…etc.) est représenté par plusieurs éléments. Un bon de commande par exemple est représenté par son numéro de commande, client, produit,…etc. Chaque élément est lui-même décrit par d'autres propriétés (produit est structuré par son numéro produit, quantité,…etc.).Et ainsi de suite ; le tout est bien sur traduit en langage de balise. D'autres informations peuvent être ajoutées à l'intérieur même des balises. Cette façon de structurer un type quelconque de document porte le nom DTD, « Définition de types de documents » ou de schémas XML.

Exemple :

Soit une commande qui est identifiées par son numéro commande « 123 », faite par le client « X » pour les produits qui possèdent comme numéro produit : « a100 »et « a102 » de quantités (respectivement) : « 1 » et « 3 ».

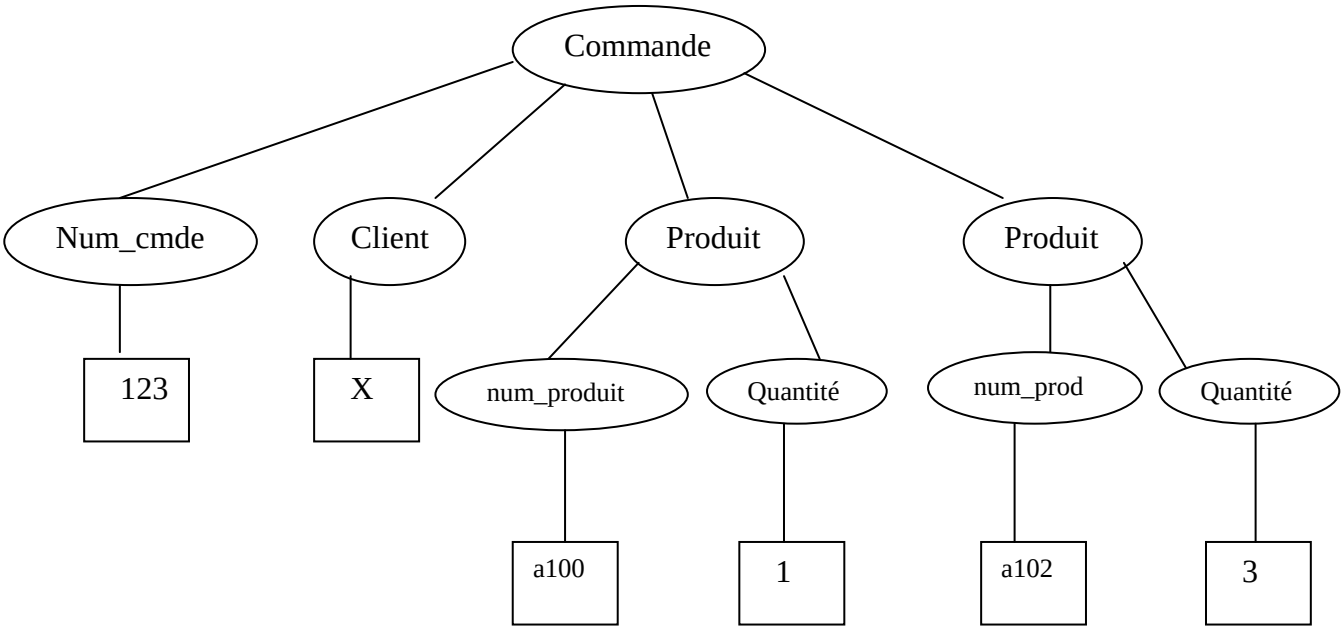


Figure I.2 : Schéma d'arborescence XML représentant une commande.

Son code XML est :

< ? xml version= "1.0" ?>
<commande num_cmde="123 " client="X">
<produit num_produit ="a100" quantite="1"/>
<produit num_produit ="a102" quantite="3"/>
</commandes>

IV.2.3. Le web sémantique et l'XML

XML reste un langage très important pour structurer le contenu du Web malgré le nombre assez réduit des documents XML actuellement disponibles. Celui-ci est utilisé sur le web car il permet la description de documents électroniques par l'intermédiaire d'une DTD.il fait le lien entre les documents et les logiciels qui les utilisent.

En effet, le but de l'XML est de faciliter la diffusion et l'échange d'informations sur Internet.

Parmi les avantages que XML pourrait apporter au web en général et au web sémantique en particulier :

- Son indépendance par rapport aux plates formes.
- Son exploitation possible par un système informatique.
- Normalisation de la représentation du contenu.
- Normalisation des données.
- C'est un langage de description facilement extensible en fonction des besoins des applications.

IV.2.4. Les avantages et inconvénients de XML

L'avantage de l'XML est la possibilité de personnaliser des documents en utilisant XSL (XML Stylsheet langage), qui permet de transformer automatiquement un fichier XML en une page HTML, qui est consultable via un navigateur Internet.

Cependant son inconvénient est que l'XML n'a pas de sémantique formelle permettant l'interprétation par la machine.XML décrit uniquement la structure de l'information (sa syntaxe).

Exemple :

Si on lance une recherche sur le web avec les mots clés "Cauchy" et "suite ", la plupart des documents retournés porteraient sur « Le mathématicien Cauchy » et « les mathématiques »Or l'objectif initial visait des informations à propos du critère de convergence des suites de Cauchy.

IV.3. RDF

RDF (ressource Description Framework) est un langage formel qui permet d'affirmer des relations entre des « ressources ».Il est utilisé pour annoter des documents écrits dans des langages non structurés, ou comme une interface pour des documents écrits dans des langages ayant une sémantique équivalente (des bases de

données ,par exemple).RDF est muni d'une syntaxe basée sur XML, et d'une sémantique.

Un document RDF est un ensemble de triplets de la forme <sujet, prédicat, objet> ; ou le sujet représente la ressource, le prédicat représente la propriété déscriptive, et l'objet représente un latéral ou une autre ressource. Cet ensemble de triplets peut être représenté de façon naturelle par un graphe orienté et étiqueté, ou les éléments apparaissant comme sujet ou objet sont les sommets, et chaque triplet est représenté par un arc dont l'origine est son sujet et la destination est son objet.

Exemple :

La représentation par un graphe RDF d'une émission de débat, dont l'animateur est « Yves Calvi », de durée de « 55 minutes », intitulée « C dans l'air », et qui est diffusée sur la chaine « France 5 » est schématisée dans la figure 3, dont les objets d'un triplet qui sont des littéraux sont représentés dans un rectangle, les ressources sont représentées dans des ellipses et les propriétés sont représentées par des étiquettes sur les arcs.

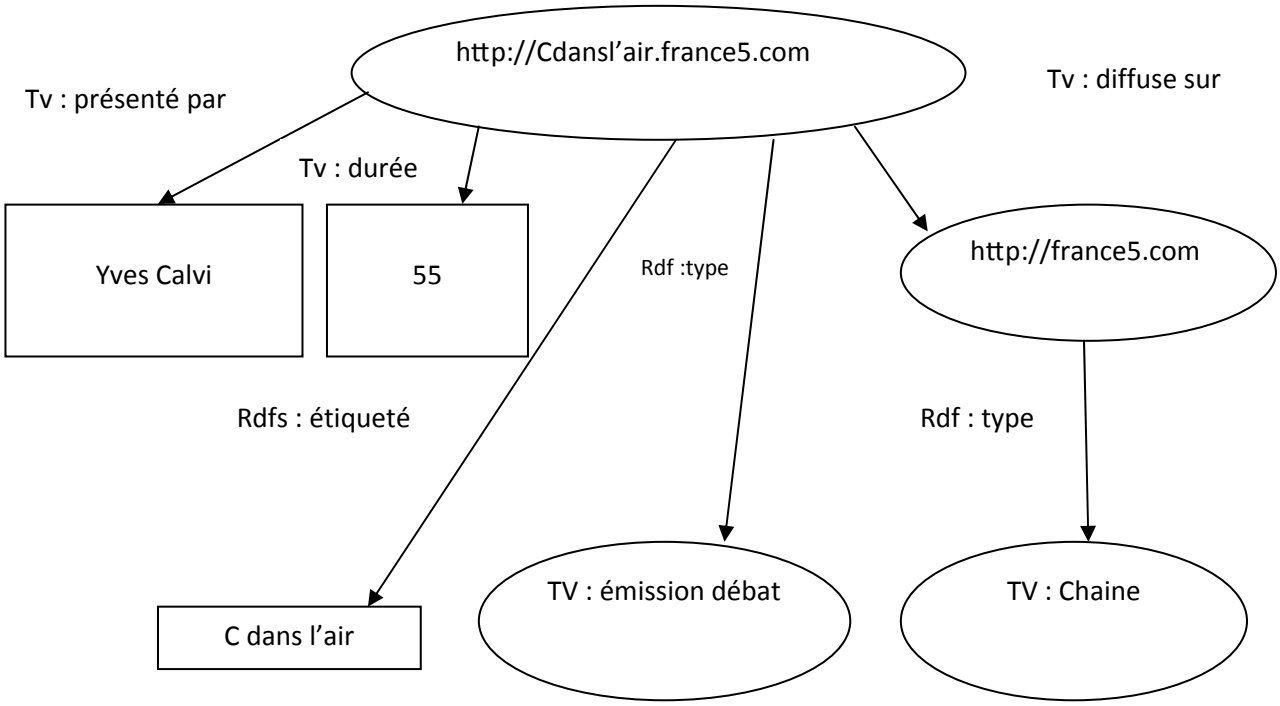


Figure I.3 : Schéma d'un graphe RDF représentant une notice documentaire.

Grace à RDF, un service effectue des affirmations prouvant que des entités possèdent des particularités propres à elle avec des valeurs bien précises. Par exemple l'entité « personne » peut posséder des propriétés comme « est le frère de », « est le détenteur de », avec des valeurs « nom d'une autre personne », «l'objet détenu ».

Cette structure s'avère la voie la plus naturelle pour décrire les données qui vont être traités par les machines. Le sujet, le verbe et l'objet sont identifiés par des URIs « Universal Ressource Identifier ».Pour créer un nouveau concept, il suffit de définir au sein du Web un nouveau URI qui fait référence à lui. Toutes les entités peuvent être pointées par les URIs. De cette manière un moteur de recherche exploitant la syntaxe RDF, rendra celle-ci plus pertinente.

IV.4. Les Ontologies

L'ontologie est une forme spécifique explicite et formelle utilisé pour la conceptualisation **[5]**.

Plusieurs définitions du mot ontologie ont été données, notamment dans **[6]** et **[7]** et dans d'autres travaux. La définition suivante : « *une ontologie est la définition de concepts, relations entre concepts, contraintes et règles d'inférences qui seront utilisées par un système de représentation des connaissances*. »Trouvée dans **[8]** sera retenu.

Les ontologies spécifient les connaissances conceptuelles explicitement, en utilisant un langage formel ou semi-formel, offrant une sémantique plus ou moins rigoureuse, permettant une utilisation non ambiguë des termes. Dans la communauté de l'ingénierie des connaissances (IC), le terme ontologie est souvent associée à un méta modèle qui décrit le contenu d'une base, ses propriétés, la manière dont elle peut être utilisée et cela en plus du vocabulaire et de la syntaxe fournis par le langage de représentation. Les connaissances intégrées dans les ontologies sont formalisées en mettant en jeu cinq types de composants : les classes, les relations, les fonctions, les axiomes et les instances. **[9]**

- **Les classes :** sont habituellement organisées en taxonomie. Une taxonomie est une hiérarchie de concepts (ou d'objets) reliés entre eux en fonction d'une sémantique particulière.
- **Les relations :** représentent un type d'interaction entre les concepts d'un domaine. Elles sont formelles définies comme tout sous-ensemble d'un produit de n ensemble, c'est à dire R : E1xE2x……xEn. Des exemples de relations binaires sont : sous classe de ; connecté à ; sorte de ; etc.
- **Les fonctions :** sont des cas particuliers de relations dans lesquelles le $n^{ème}$ élément de la relation est unique pour les n-1 précédents. Formellement, les fonctions sont définies ainsi, F : E1xE2x…xEn-1, En. Comme exemple de fonction binaire, nous avons la fonction mère de **[10]**.
- **Les axiomes** de l'ontologie permettent de définir la sémantique des termes (classes, relation), leurs propriétés et toutes contraintes. Quant à leurs interprétations, elles sont définies à l'aide de formules bien formées de la logique de premier ordre en utilisant les prédicats de l'ontologie.
- **Les instances** sont utilisées pour représenter des éléments.

Définissant aussi les concepts et les attributs.

- **Les concepts** consistent en la description d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement, etc. Ils peuvent être abstraits ou concrets, élémentaires ou composés, réels ou fictifs.

- **Les attributs** sont des restrictions des concepts ou des classes.

IV.4.1 Les types d'ontologies

IV.4.1.1Connaissances et domaine de connaissance

Une ontologie ne peut être construite que dans le cadre d'un domaine précis de la connaissance, ne serait-ce que parce que beaucoup de termes n'ont pas le même sens d'un domaine à l'autre, et qu'une sémantique non ambiguë doit être intégrée à l'ontologie. Un domaine de connaissances est donc constitué par les objets du domaine et par un contexte d'usage de ces objets **[9]**.

Délimiter rigoureusement un domaine de connaissances peut par contre se révéler ardu, à cause de la nature de la connaissance. Certaines connaissances, qui peuvent constituer en elles-mêmes un domaine, sont utilisées dans tous les autres domaines. De plus, les connaissances humaines se déploient suivant plusieurs dimensions : des connaissances peuvent être développées non seulement sur la réalité mais également sur un domaine de connaissance. On parle alors de méta connaissances, ou connaissances sur les connaissances.

Le domaine de connaissances sur lequel porte l'ontologie n'est pas le seul critère permettant de la typer. Dans **[11]**, M.Uschold considère que les ontologies varient suivant trois dimensions : le degré de formalisme de la représentation (qui varie continûment depuis le rigoureusement formel jusqu'au hautement informel), l'objectif opérationnel (communication entre utilisateurs, interopérabilité entre systèmes, application à un problème d'ingénierie comme la réutilisation de composants, résolution de problèmes) et le sujet (domaine de connaissances, connaissances et raisonnement, connaissances liées au modèle de représentation). Ainsi une ontologie est dite :

1. **hautement informelle (highly informal),** dans le cas d'utilisation du langage naturel sans aucune restriction.
2. **semi-informelle (semi-informal),** lorsque le langage est de type langage naturel mais structuré avec un vocabulaire limité afin de restreindre les ambiguïtés.

3. **Semi-formel (semi-formal)** si l'ontologie est représentée à l'aide d'un langage artificiel défini de façon formelle.
4. **rigoureusement formelle (rigorously formal**) lorsque les termes possèdent une sémantique dans un système tel que le calcul des prédicats du premier ordre. L'ontologie TOVE en est un exemple.

Dans la deuxième classe, l'ontologie se définit selon le domaine étudié et le degré de généralité ou de précision des connaissances représentées. Les types d'ontologies les plus utilisées sont : les ontologies génériques, les ontologies du domaine et les ontologies d'application.

IV.4.1.2 Les concepts et les relations

Les connaissances portent sur des objets auxquels on se réfère à travers des concepts. Un concept peut être divisé en trois parties : un terme (ou plusieurs), une notion est un ensemble d'objets. La notion, également appelé extension du concept, regroupe les objets manipulés à travers le concept, ces objets sont appelés instances du concept.

Par exemple, le terme « *élément de connaissance* » renvoie à la fois à la notion de l'élément de connaissance comme objet de type « granule de connaissance », et à l'ensemble des objets de ce type.

Il est à noter que :

- Un concept peut très bien avoir une extension vide (concept générique).
- Deux concepts peuvent partager la même extension sans pour autant avoir la même intension.
- Les concepts partageant la même extension mais pas leur intension peuvent être désignés par le même terme.

Les propriétés suivantes sont nécessaires à la complétude de la définition du concept.

Ces propriétés sont mieux expliciter dans la page suivante :

- ***La généricité :*** un concept est générique s'il n'admet pas d'extension. e.g : la vérité est un concept générique.
- ***L'identité :*** (propriété proposée par N.Guarino). Un concept porte une propriété d'identité si cette propriété permet de conclure quant à l'identité de deux instances de ce concept.
- ***La rigidité :*** (propriété proposée par N.Guarino). Un concept est rigide si toute instance de ce concept en reste instance dans tous les mondes possibles. e.g: personne est un concept rigide, étudiant est un concept non rigide.
- ***L'anti-rigidité :*** (propriété proposée par N.Guarino). Un concept est anti-rigide si toute instance de ce concept est essentiellement définie par ses appartenances à l'extension d'un autre concept. e.g : étudiant est un concept anti-rigide car l'étudiant est avant tout une personne.
- ***L'unité :*** (propriété proposée par N.Guarino). Un concept est un concept unité si, pour chacune de ses instances, les différentes parties de l'instance sont liées par la relation qui ne lie pas d'autres instances de ce concept.

Les propriétés portant sur deux concepts sont :

- **L'équivalence :** deux concepts sont équivalents s'ils ont même extension.
- **La disjonction :** (on parle aussi d'incompatibilité). Deux concepts sont disjoints si leurs extensions sont disjointes.
- **La dépendance :** (propriété proposée par Guarino). Un concept C1 est dépendant d'un concept C2 si pour toute instance de C1 il existe une instance de C2 qui ne soit ni partie ni constituant de l'instance de C2.

Tout comme les concepts, les relations peuvent être spécifiées par des propriétés dont une liste, non exhaustive, est donnée ci après. Les relations sont organisées de manière hiérarchisée à l'aide de la propriété de subsomption (un concept C1 subsume un concept C2 si toute propriété sémantique de C1 est aussi une propriété sémantique de C2, c'est-à-dire que si C2 est plus spécifique que C1)décrite précédemment. Les

remarques faites au sujet de l'organisation des propriétés s'appliquent également dans ce cas.

- **Les propriétés algébriques:** symétrie, réflexivité, transitivité.
- **La cardinalité:** nombre possible de relations de ce type entre les mêmes concepts (ou instances de concept).

Les propretés liant deux relations sont :

- **L'incompatibilité :** deux relations sont incompatibles si elles ne peuvent lier les mêmes instances de concepts.
- **L'inverse *:*** deux relations binaires sont inverses l'une de l'autre si, quand l'une lie deux instances I1 et I2, l'autre lie I2 et I1.
- **L'exclusivité :** deux relations sont exclusives si, quand l'une lie des instances de concepts, l'autre ne lient pas ces instances, et vice-versa. L'exclusivité entraîne l'incompatibilité.

Les propriétés liant une relation et des concepts sont :

- **Le lien relationnel :** Il existe un lien relationnel entre relation R et deux concepts C1 et C2 : il existe une relation de type R qui lie les deux instances de C1 et C2. Un lien relationnel peut en outre être contraint par une propriété de cardinalité, ou porter directement sur une instance de concept **[12]**.
- **La restriction de relation :** pour tout concept de type C1, et toute relation de type R liant C1, les autres concepts liés par la relation sont d'un type imposé.

IV.4.2. Les langages de représentation d'ontologies

Un langage formel (ou formalisme) est un ensemble de composants sémantiques (contenu), de règles structurelles (mode d'emploi) et d'une notation formelle particulière (forme) destinée à organiser la définition ainsi que les relations entre les éléments constitutifs d'un système. L'objectif de l'utilisation d'un langage de représentation d'ontologies de haut niveau, est de permettre de réduire les ambiguïtés de la langue naturelle en offrant une plus grande déclarativité. Le langage utilisé pour décrire les termes d'une ontologie a un impact direct sur son niveau de formalisme. Les formalismes offrent un support formel à la composition des concepts et à leur comparaison **[13].**

Certains langages, comme les logiques de description, offrent des mécanismes d'inférence qui permettent de vérifier la consistance des termes et d'inférer de nouvelles connaissances. Il existe alors, de nombreux langages de représentation qui peuvent être regroupés en : langages de type logique permettant de construire des ontologies rigoureusement formelles ; et langages à base de schémas (frames) qui permettent de modéliser directement les concepts et les objets.

Nous définissons les particularités de chacune des deux catégories ensuite, nous présentons quelques langages.

IV.4.2.1 Les langages de type « *logique* » (logique terminologique)

L'objectif est d'utiliser des langages reposant sur une formalisation logique. Citons les langages basés sur les logiques descriptives (logiques terminologiques) tels que KL-One et Classic, qui permettent de représenter des concepts automatiquement organisés en une taxonomie, des conditions nécessaires et suffisantes pour l'appartenance à une classe, ainsi que la détection d'incohérences. CycL est un langage d'expression de l'ontologie Upper Cyc dont la syntaxe décrive de la logique du premier ordre et de Lisp. Citons également, les graphes conceptuels de Sowa qui sont des réseaux sémantiques partitionnés, permettant de définir de façon plus souple

la représentation de significations de termes tout en autorisant une formalisation logique.

IV.4.2.2. Le langage de type « *frame* »

L'ontologie obtenue est moins rigoureuse qu'avec un langage de type logique, tout en offrant une formalisation. Ces langages vont du moins structurés (réseaux sémantiques, ensemble de nœuds reliés par des arcs étiquetés,…), aux langages dits de « schémas » ou « frames ». Les langages à base de schémas sont nombreux et possèdent une puissance d'expression intéressante. Nous pouvons classer dans cette catégorie Loom qui est à la fois un environnement et un langage de programmation dédiés à la construction de bases de connaissances. Ecrit en CommonLisp, Loom propose un langage de description pour la modélisation de concepts, d'objets et de relations.

Les langages les plus couramment utilisés sont : Ontolingua, CycL, Loom, Flogic **[14]**. Ils utilisent une approche qui se base sur les frames, sur la logique des prédicats ou sur les deux.

Ontolingua : Est un langage basé sur KIF *(*Knowledge Interchange Format*)* et sur la Frame Ontology. KIF intègre une sémantique déclarative, basée sur l'analyse des prédicats de premier ordre, à une puissance d'expressivité suffisante pour représenter les connaissances contenues dans la base de connaissances du système des applications. Frame Ontology est moins expressive que KIF mais elle permet de représenter l'ontologie par des termes tels que classe, instance, sous classe de, etc.

CycL : Est un langage de représentation de connaissances de Cyc. Ce langage, déclaratif et expressif proche de la logique des prédicats, comprend des extensions qui permettent de traiter l'égalité, le raisonnement par défaut, l'application de la fonction de Skolem et quelques aspects de la logique de second ordre.

Loom : Est un langage de programmation perfectionné basé sur la logique et l'environnement de premier ordre. Il fournit un langage expressif et explicite de spécification de modèles déclaratifs ainsi qu'un support déductif puissant. Il offre plusieurs paradigmes de programmation qui constituent une sorte d'interface avec la spécification de modèles déclaratifs, et des services à base de connaissances.

Flogic : Est une intégration de langages de frames et de logique des prédicats. Il inclut des objets (simples et complexes), le principe d'héritage, les types polymorphes, les méthodes de recherche et l'encapsulation. Il contient des règles permettant d'inférer de nouvelles connaissances à partir des connaissances existantes.

A coté de ces langages représentatifs et stables, d'autres langages se sont développés. Une étude comparative recense quelques langages candidats à l'écriture d'ontologies **[10]**.

OML/CKML : Le but de OML (Ontology Markup Language) est de représenter les ontologies par la syntaxe de XML. CKML (Conceptual Knowledge Markup Language) assure un cadre conceptuel pour la représentation des connaissances réparties, basé sur les graphes conceptuels.

XML/RDF : RDF (Resource Description Framework) se base sur un modèle mathématique de graphes orientés et étiquetés. Les classes sont organisées hiérarchiquement en utilisant les classes et sous-classes qui forment le standard d'héritage et les classes dans le modèle objet.

UML : Assure un ensemble de notations conventionnelles qui permettent aux concepteurs des logiciels de modéliser leurs systèmes.

OKBC : Open Knowledge Based Connectivity est une interface de programmation d'applications (API), qui permet d'accéder au cadre de représentation des connaissances. Son but est d'assurer un modèle uniforme qui pourra être compris par un nombre de systèmes de représentation des connaissances.

IV.4.3. Les ontologies et le Web sémantique :

Les ontologies jouent un rôle important pour faire communiquer les personnes et les machines dans le Web sémantique. En effet Une fois construite et adoptée par une communauté particulière, une ontologie doit traduire un certain consensus explicite et certain niveau de partages qui sont essentiels pour permettre l'exploitation de ressources sur le Web.

La sémantique du Web est fondée sur des ontologies spécifiées explicitement dans un langage de représentation. Le W3C cherche à proposer un standard connu actuellement sous le nom d'OWL(Ontology Web Langage),qui s'appuie sur la logique de description.

IV.5. Logique et inférence

Utiliser pour amorcer des inférences et pour pouvoir les expliqués (exprimer des règles de raisonnements dans des systèmes a base de connaissances.

IV.6. Preuve et confiance

Le Web étant un espace d'information, garantir la fiabilité et l'origine des informations utilisées doit reposer sur des méthodes de qualification de l'origine de l'information, par exemple par des méta-données et des annotations, éventuellement certifiées par des signatures électroniques.

V. Composants principaux du Web sémantique [15]

Le Web sémantique s'articule autours de deux composants essentiels :

1. **Les ontologies :** technologie dorsale pour le Web sémantique et – plus généralement - pour le management des connaissances formalisées décrivant les ressources du Web.

Elles fournissent la "sémantique" exploitable par machine des données et des sources d'informations qui peuvent être communiquées entre différents agents (logiciel et humaines).

2. **Les annotations sémantiques :** décrivent les ressources en utilisant la "sémantique" définie dans l'ontologie.

Les ressources annotées par les métadonnées faciliteront la recherche, l'extraction, l'interprétation et le traitement de l'information d'une manière plus efficace.

V. Annotation et métadonnées dans le web sémantique [16]

Un des grands principes du Web sémantique est qu'il est nécessaire d'associer aux ressources du Web des informations exploitables par des agents logiciels afin de favoriser l'exploitation de ces ressources.

Associer par exemple une notice comprenant des champs : Auteur, Date de création, Date de modification, Mots-clés, à une page Web permet de considérer celle-ci non plus seulement comme comprenant du texte, mais également des informations structurées à la sémantique connue et utilisable comme telle par un agent logiciel.

Associer une information exploitable à une ressource signifie deux choses essentielles :

- La première est que cette information doit d'une manière ou d'un autre être structuré – utilisable – et descriptive – de la ressource, de son utilisation. Il s'agit d'en faciliter et améliorer l'accès dans le cas d'une ressource directement visualisée par un utilisateur et permettant une recherche d'information plus efficace et plus ciblée.

- La seconde est que la ressource en question doit exister et pouvoir être exploitée sur le Web indépendamment des informations qui lui sont associées dans le cadre du Web sémantique : celles-ci sont utiles, mais non nécessaires pour accéder et utiliser la ressource, la page Web ou le service.

VII. Les langages du Web sémantique [17]

Le Web sémantique doit pouvoir être manipulé par les machines. Il est alors nécessaire de disposer de langages pour :

- Exprimer les données et les métadonnées.
- Exprimer les ontologies.
- Décrire les services.

Il existe déjà des langages développés pour ces activités indépendamment du Web sémantique.

Cependant, ils ne sont pas utilisés tels quels dans le Web sémantique car il est nécessaire de leur permettre d'accepter les caractères propres au Web à savoir sa distribution (il faut être capable de tirer parti de l'information dont on ne dispose pas localement) et son ouverture.

Il semble clair que le Web sémantique ne pourra voir le jour sans un minimum de standardisation. Différents consortiums et organismes mettent donc les acteurs autour d'une table pour définir les langages à utiliser dans le Web sémantique.

L'intérêt de cette approche standardisantes est bien sûr d'assurer des traitements uniformes sur l'ensemble des documents écrits dans ces langages qui permettront diverses applications nouvelles telles que :

- La recherche d'information fondée sur des descriptions formelles ;
- La composition de services en fonction de leurs descriptions
- L'interconnexion de catalogues sur la base de leur description.

Le but du Web sémantique est principalement que les services soient mieux rendus sans engendrer de surcharge pour les utilisateurs. Dans cette perspective, les usages ne devraient se voir impacter que positivement par les langages développés. Mais l'idée du Web est que les usagers en soient les contributeurs. C'est en ce sens que les langages développés pour le Web sémantique ont un impact sur ceux qui les utilisent pour décrire leurs ressources, voire leurs services.
Les travaux de standardisation sont aujourd'hui bien avancés : RDF, OWL et SOAP sont des recommandations du W3C et TopicMaps une norme ISO.

Nous décrirons ici trois sortes de langages :

- Des langages d'assertions (RDF et cartes topiques) ;

- Un langage de définition d'ontologies pour le Web (OWL) ;
- Différents langages de description et de composition de services (UDDI et autres).

Les deux premiers cas s'articulent autours de langages proposés par le W3C .Ces langages sont munis d'une sémantique formelle en théorie des modèles. Un des intérêts de munir les langages d'une sémantique formelle est de pouvoir définir de façon naturelle la notion de conséquence : un document RDF est conséquence d'un autre veut dire que toute information contenue dans ce dernier est aussi contenue dans le premier ; et une classe OWL est conséquence d'une autre veut dire que toutes les instances de la seconde sont des instances de la première. Ceci permet de comparer des faits (dans RDF) ou des classes (dans OWL), et permet donc d'interroger une base de documents : l'utilisateur peut par exemple définir un document RDF (la question), et lancer un mécanisme de recherche sur le Web pour les documents RDF dont la question est une conséquence. Ce seront les réponses à cette question. Ceci ne fait cependant pas de RDF un vrai langage de requêtes. Bien qu'il puisse répondre à certaines questions (« y a-t-il un train de Grenoble à Paris partant entre 8h00 et 9h30 demain ? »), il ne permet pas d'agir (dans ce cas, réserver les billets). Il faudrait pour cela encapsuler ces langages dans des langages de requêtes similaires à ceux que l'on peut trouver en bases de données, à moins que cette tâche ne soit totalement dévolue aux services.

VIII. Conclusion

Dans ce chapitre une vision globale du Web sémantique a été donnée, une étude de la plus basse à la plus haute couche le composant a été établie, explicitant les éléments ainsi que les langages qui régissent son fonctionnement. L'objectif final de ce travail est de concevoir et de développer des techniques permettant de mettre en correspondances des communautés, individus, activités, etc… dans un réseau social.

Le prochain chapitre aura pour objectif de définir cette nouvelle génération d'outils destinés à la communication humaine et d'étudier ces application et services du Web qui visent à l'accroissement de la conscience mutuelle entre les individus.

I. Introduction [18]

Avec le développement des nouvelles technologies du Web 2.0, il est aujourd'hui matériellement possible de répondre aux besoins de communication d'un réseau.

Ces nouvelles applications sont capitales pour permettre un travail collaboratif entre un groupe de personnes souvent séparées géographiquement voire temporellement.

II. Le Web Social [19]

Le Web Social est né de l'apparition d'une nouvelle génération d'outils destinée au soutien de la communication humaine. Il correspond à un ensemble d'applications du Web qui visent essentiellement à fournir des espaces de rencontres entre membres de communautés différentes, accroissant ainsi la conscience mutuelle entre les individus.

A L'appellation Web Social (WSo) est directement liée au terme de **logiciel social** ou **relationnel**. L'objectif du WSo est la communication pour favoriser les liens sociaux au travers d'échanges, de discours. Le web social prône l'utilisation d'outils s'articulant autours d'une informatique communicante donc dont la formalisation est peu contraignante et d'une grande richesse sémantique.

Le réseau du Web Social est un réseau social.

III. Les Réseaux sociaux [3]

Les systèmes de recherche d'information classiques se fondent principalement sur une architecture centralisée ou l'utilisateur exploite un service de recherche, publique ou communautaire, afin de trouver les informations correspondant à son intérêt. De tels services sont fournis par exemple par les portails sur l'Internet, soit pour la recherche publique soit pour le besoin d'une entreprise ou une communauté.

Dans ce contexte les utilisateurs ne jouent que le rôle des consommateurs du service dans la recherche, aucune collaboration n'est nécessaire entre eux.

Dans la société de l'information à venir, le rôle humain reste important mais il le deviendra de plus en plus. Une personne ne sera plus seulement traitée comme un consommateur mais aussi comme une source de connaissances dans la mesure où elle collectionnera et qualifiera un ensemble d'informations selon son intérêt sans sa propre base de connaissances. Une telle base de connaissances pourra jouer alors le rôle d'un intermédiaire dans la recherche d'information par d'autres personnes. Ainsi on voit apparaitre une approche de recherche d'information qui exploite la collaboration entre les personnes dans un réseau social. Dans cette nouvelle approche, la recherche d'information est une pratique collective. Les utilisateurs participeront eux-mêmes aux processus de valorisation d'informations dans la recherche. Cette valorisation sera transformée sous forme de connaissances individuelles et partagées avec les autres membres de la communauté. Une convergence de trois principes est envisagée dans une telle approche : **filtrage collaboratif**, **répartition de connaissances** et **centrage sur les personnes**.

III.1. Filtrage collaboratif

Le filtrage collaboratif ,en principe ,se base sur l'hypothèse que les gens occupés à la recherche d'information devraient pouvoir se servir de ce que d'autres ont déjà trouvé et évalué**[20].**Nous considérons que les appréciations/annotations des utilisateurs sur les informations trouvées sont modélisées comme des connaissances dans leur mémoire personnelle(dépendant de leur profil d'utilisateur).Pour chaque utilisateurs ,un ensemble de proches voisins est identifié à travers un réseau social, qui est soit calculable par un automatisme soit établi par un facteur humain. Ainsi la quantité et qualité d'informations, trouvées par une personne ayant lancé une recherche, dépendront des documents qualifiés par son voisinage.

Il existe effectivement deux approches utilisées dans les systèmes de filtrage d'informations collaboratifs. La première : **filtrage automatisé**, emploie des méthodes statistiques pour faire des prévisions basées sur des configurations des intérêts des utilisateurs. Les utilisateurs fournissent alors des évaluations des documents sous forme de notes, pour constituer leur profil. Ces estimations sont comparées à celles d'autres utilisateurs et les similitudes sont mesurées. Des prévisions sont calculées comme moyenne pondérée des avis d'autres utilisateurs avec des gouts soit semblables, soit complètement opposés. Ces prévisions sont ainsi exploitées pour faire des propositions à un individu sur ce qui a été apprécié par des personnes dont les gouts sont proches des siens.

L'autre motivation pour le filtrage collaboratif, appelée « recommandation active » **[21],** vient d'une pratique courante chez les utilisateurs ou on envoie des pointeurs sur des documents intéressant à des collègues ou des amis. Cette fonctionnalité peut être intégrée à un système de recherche d'information et permet à ses utilisateurs d'adresser des pointeurs aux personnes qu'ils jugent intéréssées.Nous suivrons cette approche pour construire un réseau d'échanges entre utilisateurs. Chaque utilisateur garde dans sa mémoire les contacts possibles avec des « proches » afin de faciliter ses échanges actifs de pointeurs.Lorqu'un document est qualifié, un pointeur sur celui-ci est adressé automatiquement aux personnes qui sont restées en contact.

III.2. Répartition de connaissances

Opposée à la technique d'indexation dans les portails classiques qui reposent principalement sur un entrepôt de métadonnées, la recherche d'information dans un réseau social requiert une exploitation de multiples bases de connaissances. Le terme de métadonnée prend bien en compte la notion d'ajout d'information à une ressource, et on pourra a priori les utiliser indifféremment pour rendre celle-ci plus pertinente dans la recherche d'information. Chacun des utilisateurs dans le réseau social possède une base de connaissances personnelle pour stocker des métadonnées valorisées par

lui-méme.Une telle base de connaissances est partagée avec les autres utilisateurs sous contrôle de son propriétaire afin de faciliter l'accès collectif à l'information. Cette architecture distribuée pourrait procurer l'autonomie maximale de chacun des participants dans la communauté.

III.3. Centrage sur les personnes

Les systèmes d'information ou plates formes développés pour un support communautaire, reposent en pratique sur l'exploitation d'une ontologie standarisée.Cette ontologie impose de fait un schéma commun pour les connaissances fournies par différents utilisateurs .Ceci implique qu'il n'y a plus d'hétérogénéité dans l'échange de connaissances entre utilisateurs. Néanmoins, cette approche simple exige toujours des efforts important de construction et d'entretiens de d'ontologie commune parfois très complexe. De plus, elle peut apparaitre encombrante et inflexible une fois appliquée à un problème personnel et spécifique. Donc au lieu de forcer l'existence d'une ontologie commune, nous suivrons l'approche dans laquelle chacun peut définir sa propre ontologie, mais en la reliant avec celles définies par d'autres à travers des « colles sémantique ».Cette approche « bottom-up »,d'une part semble bien adapté à la grande échelle d'un système global comme le Web, et d'autres part, met en pratique une compensation entre la flexibilité et la réutilisation dans l'ingénierie des connaissances.

IV. Bookmarks partagés

Avec l'émergence des signets ou marque-pages, les navigateurs actuels offrent aux utilisateurs un moyen limité pour collectionner et organiser personnellement des références, via leur URL, à des pages web choisies. Le besoin de partage de bookmarks apparait rapidement entre utilisateurs ayant les mêmes intérets, par email ou par un service approprié qui leur permet de s'échanger les URL de ressources intéressantes.

Plusieurs formats de données (XML, HTML) peuvent être utilisés pour faciliter ces échanges de bookmarks partagés **[22]**.

En effet, Annotea est un serveur RDF qui permet aux utilisateurs connectés au réseau de partager leurs annotations ainsi que leurs signets, sur des ressources du web. En utilisant un client, un utilisateur peut trouver visualiser des signets, liens partagés sur ce serveur.

Notons que :

- Le partage de liens signets ne se limite pas aux serveurs centralisés, mais peut être appliqués dans un environnement pair-à-pair.

- Le filtrage collaboratif peut être appliqué également aux signets partagés.

Sisteseer **[23]** est un système de recommandation de pages web qui utilise les bookmarks (signets) personnels et leur organisation en répertoire pour prédire et recommander des pages pertinentes. Il considère chaque répertoire de signets d'un utilisateur comme une déclaration implicite d'intérêt. Il consulte alors les signets de chaque utilisateur et mesure le degré de chevauchement (nombre d'URL communes) de chaque répertoire avec les répertoires d'autres utilisateurs.

Le système ne tire aucune sémantique ni du contenu des URL ni du répertoire dans cette mesure de chevauchement .En effet, le chevauchement permet de déterminer les similarités entre répertoires et ainsi de former dynamiquement des communautés virtuelles.

Dans tous les cas les bookmarks (signets) ont des limitations spécifiques **[20]**.Tout d'abord, les utilisateurs ne marquent pas souvent des sites qu'ils trouvent intéressants parce qu'un site peut être accessible à travers d'autres chemins (via un hyperlien ou un moteur de recherche).

Dans l'usage de signets, il est difficile de connaitre la raison pour laquelle une personne a marqué une page. C'est peut-être un véritable intérêt ou simplement un besoin de revisiter ou d'y retourner.

V. Réseaux sociaux et Web sémantique [24]

Les interactions des utilisateurs au travers des usages du web 2.0 amènent la communauté à réfléchir sur les moyens de capter ces usages pour y appliquer les techniques d'analyse des réseaux sociaux.

Les applications bien connues à l'origine de l'émergence du web 2.0 sont les blogs, les wikis (ex : wikipedia), les services de social bookmarking (ex : del.ico.us), les sites de partages de médias (ex : youtube, flickr) et bien sûr les sites de réseaux sociaux (ex : facebook, LinkedIn).

Ces applications ont considérablement accru la participation, les interactions et le partage entre les utilisateurs du web.

L'analyse et la compréhension de tels réseaux sociaux suscitent de vifs intérêts au sein de plusieurs communautés scientifiques.

Le web sémantique fournit des formalismes pour la représentation sémantique des personnes et de leurs usages sur le web.

Exemple :

- L'ontologie **FOAF** décrit "les personnes, les liens entre elles, ce qu'elles créent et ce qu'elles font".

- L'ontologie **SIOC** décrit "l'information contenue explicitement et implicitement dans les moyens de communication d'internet" comme, par exemple, les blogs.

- L'ontologie **SCOT** est un moyen de "représenter la structure et la sémantique des données du social tagging afin de les partager et de les réutiliser".

En regard de ces moyens de représentation il existe un certain nombre de propositions d'utilisation des méthodes d'analyse des réseaux sociaux pour extraire des informations. La plupart de ces méthodes d'analyses sont basées sur la théorie des graphes.

Exemple :

- **[25]** exploite certaines propriétés de la théorie des graphes afin d'identifier des champs sémantiques et des communautés d'intérêt.

- Les travaux de **[26], [27]** ont permis d'extraire des chemins entre des ressources sémantiquement liées dans les graphes RDF, fournissant ainsi une base pour une représentation et une analyse sémantique d'un réseau social.

VI. Représentation d'un réseau social

La première personne à avoir représenté un réseau social est Jacob Levy Moreno au début des années 1930 **.**Son objectif étant de visualiser graphiquement un réseau social, en représentant les personnes par des points et une relation entre deux personnes par des flèches.

Cette représentation est depuis désignée par le terme sociogramme, mais on parle également de toiles en raison de leur aspect en toile d'araignée. Cette forme de visualisation, fut un premier outil d'identification rapide des caractéristiques d'un réseau social.

Moreno a ainsi introduit le concept d'étoile pour désigner les personnes ayant le plus de relations dans un réseau social, en référence à l'étoile formée par un point et ses connections.

Une approche mathématique a vite été établit du fait du rapprochement entre la théorie des graphes et représentation en sociogrammes.

Le graphe est devenu par la suite la représentation adoptée par toutes les sciences manipulant l'analyse des réseaux sociaux, dont la sociologie, les mathématiques et l'informatique.

VI .1. Théorie des graphes et réseaux sociaux

Les définitions ci-dessous listent quelques notions manipulées par la théorie des graphes pour les réseaux sociaux:

- **Un sommet** est l'unité de base d'un réseau, il en représente une ressource. Dans un réseau social on parle d'acteur. Le terme nœud est également utilisé pour désigner un sommet.
- **Une arête** est une connexion entre deux sommets. On parle également d'arc ou de lien.
- **Une hyperarête** est une arête qui connecte 2 ou plusieurs sommets.
- **Une arête** est **orientée** si elle ne s'utilise que dans une seule direction. Inversement, on parle d'arête **non orientée** pour une arête qui s'utilise dans les deux directions.
- Une arête est **pondérée** lorsqu'on lui attribue un poids.
- Une arête est étiquetée lorsqu'on lui attribue un label.
- Un **graphe** est défini par un ensemble de sommets et un ensemble d'arêtes.
- Un **graphe orienté** désigne un graphe avec des arêtes orientées.
- Un **graphe pondéré** désigne un graphe avec des arêtes pondérées.
- Un **graphe étiqueté** désigne un graphe avec des arêtes étiquetées.
- Le **degré** d'un sommet est le nombre de ses arêtes adjacentes.
- Un **chemin** est une séquence d'arêtes qui relie deux sommets.
- Un chemin orienté est une séquence d'arêtes qui relie deux sommets en respectant l’orientation du parcours à chaque arrête.
- Une **géodésique** est l'un des plus courts chemins entre deux sommets donnés.

- Le **diamètre** d'un graphe est le plus long chemin géodésique de ce graphe.
- Un graphe est **complet** lorsqu'il existe une arête entre toute paire de sommets.
- Un graphe est dit **connexe** lorsqu'il existe un chemin entre toute paire de sommets.
- Notons un graphe *G = (V, E)* avec *V* l'ensemble des sommets, *E* l'ensemble des arêtes, *n*=|*V*| et le nombre de sommets et *m*=|*E*| et le nombre d'arrêtes.
- Un sous graphe de *G* est noté *G' = (V', E')* avec *V'*Ì*V*, *E'*Ì*E* et restreint à des arrêtes reliant des sommets de *V'*, *n'*=|*V'*| et *m'*=|*E'*|.*vi* désigne le ième sommet.
- *(vi,vj)* désigne une arête entre les sommets *vi* et *vj*.
- Le degré d'un sommet *vi* est noté *ki*.

Exemple de représentation d'un réseau social :

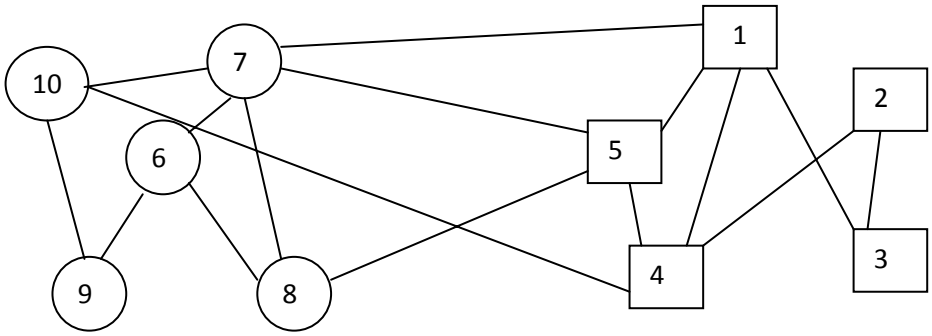


Figure II.1 : Une communauté s'est divisée en deux suite à une divergence d'opinion, les membres de la première sont représentés par des ronds et les membres de la seconde par des carrés.

.

Notons que :

- Les graphes non orientés sont adaptés pour les réseaux sociaux avec des relations non orientés.

- Les graphes orientés sont adaptés pour représenter des relations non symétriques comme les réseaux de confiance par exemple.

- Les graphes pondérés sont adaptés aux réseaux sociaux qui contiennent différents niveau d'intensités dans les relations.

- Les graphes étiquetés permettent de représenter différents types de relations.

- Les graphes multipartites sont adaptés pour des réseaux sociaux incluant différent types de ressources manipulées par les acteurs et qui sont le support d'interactions.

VI.2. Approche matricielle pour la représentation :

La matrice est l'objet mathématique le plus utilisé pour manipuler ces concepts, mais des approches ensemblistes ont aussi était proposées **[28]**.

On distingue deux types de matrices dans un réseau social :

- Les matrices d'incidence.
- Les matrices d'adjacence.

VI.2.1. La matrice d'incidence :

Les matrices d'incidence contiennent deux types de ressources, les lignes représentent un type et les colonnes un autre type.

Une matrice d'incidence est convertible en deux matrices d'adjacence représentant chacune les ressources des lignes et des colonnes (tableau II.2 et II.3), les valeurs des cases contiennent les points communs entre les ressources correspondantes dans la matrice d'incidence, *aii* n'ayant pas de valeur.

Exemple :

	Projet 1	Projet 2	Projet 3	Projet 4
Employé 1	1	1	1	0
Employé 2	1	0	0	0
Employé 3	1	1	1	1
Employé 4	0	0	1	1

Tableau II.1: Exemple de matrice d'incidence indiquant sur quel projet travaille chaque employé.

VI.2.2. La matrice d'adjacence :

On parle de matrice d'adjacence lorsqu'on a les mêmes ressources en ligne et en colonne, on obtient ainsi une matrice carrée avec la ligne *i* et les colonnes *i* représentant la même ressource.

Un graphe peut ainsi être représenté sous la forme d'une matrice *M* à *n* lignes et *n* colonnes représentant un tableau. Chaque case de ce tableau est notée *aij* avec *i* et *j* les numéros respectifs de ligne et de colonne de la case.

La valeur contenue dans la case *aij* est le poids de la relation entre les ressources *vi* et *vj* (égal à 1 dans le cas d'un graphe non pondéré), 0 correspond à une absence de relation.

Exemple 2 :

	Employé 1	Employé 2	Employé 3	Employé 4
Employé 1	-	1	3	1
Employé 2	1	-	1	0
Employé 3	3	1	-	2
Employé 4	1	0	2	-

***Tableau II.2*: Matrice d'adjacence des employés déduite du tableau II.1, chaque case représente le nombre de projets partagés entre les employés correspondants.**

Exemple 3 :

	Projet 1	Projet 2	Projet 3	Projet 4
Projet 1	-	2	2	1
Projet 2	2	-	2	1
Projet 3	2	2	-	2
Projet 4	1	1	2	-

Tableau II.3: Matrice d'adjacence des projets déduite du tableau II.1, chaque case représente le nombre d'employés partagés entre les projets correspondants

VII. Structure et aspect fonctionnel d'un réseau social :

Les principaux indices fournissant des informations importantes sur la structure et l'aspect fonctionnel d'un réseau social sont :

- La densité
- La centralité
- La résistance du réseau
- La détection des communautés.

Ces points sont plus explicités la page tournée.

VII.1. La Densité :

Indique la quantité de liens au sein d'un réseau et permet de définir la cohésion d'un réseau social. Cette mesure peut-être utilisée dans l'optique d'une analyse socio-centrée ou égocentrée Une analyse centrée sur l'individu consiste à mesurer la densité des liens autour d'un nœud donné **[29].**

La gestion des relations sociales est consommatrice en temps, ainsi le temps limite le nombre de contacts qu'une personne peut conserver et plus un réseau social est grand, moins la densité est élevée.

La densité varie également en fonction du type de relations considérées dans un réseau social, un réseau basé sur des relations amoureuses est beaucoup moins dense qu'un réseau de relations professionnelles notamment en raison des caractéristiques des liens (ex : nature exclusive, différence de temps ou de ressources requis pour l'entretien, etc.).

Ainsi le typage des relations dans un réseau social permet de paramétrer la densité.

VII.2. La centralité :

La centralité permet de détecter les positions stratégiques dans un réseau social. Il est nécessaire donc, de définir ce qui rend un nœud plus central qu'un autre, on parle alors de centralité locale. Plusieurs approches ont été considérées en fonction du critère choisi pour considérer un nœud comme plus central qu'un autre.

On en retiendra trois principales : **[29]**

VII.2.1. La centralité de degré [30]

Considère comme centraux les nœuds qui possèdent les degrés les plus élevés du graphe. En effet, ces nœuds suscitent un grand intérêt, sont très visibles, et ont un potentiel élevé à faire circuler l'information, par leur forte connectivité aux autres éléments du réseau.

VII.2.2. La centralité d'intermédiarité [29]

Se concentre sur la capacité d'un nœud à servir d'intermédiaire dans un graphe. Un nœud situé sur un chemin géodésique possède une position stratégique dans la cohésion d'un réseau et dans la circulation de l'information, d'autant plus si ce chemin est unique. Par exemple, un nœud situé sur l'unique chemin reliant deux ensembles connectés de nœuds possède un fort contrôle sur la communication de ces deux groupes. Plus un nœud est intermédiaire, plus le réseau est dépendant de lui et plus il a de pouvoir.

VII.2.3. la centralité de proximité [29]

Mesure la centralité d'un nœud en se basant sur la taille des chemins qui le lient aux autres nœuds. Cette mesure représente la capacité d'un nœud à se connecter rapidement avec les autres nœuds du réseau.

La **centralité globale**, ou centralisation, d'un réseau social est calculée à partir des centralités locales des sommets. L'indice de centralité locale choisi détermine le sens de la centralité globale. Le calcul de la centralisation dépend de la définition de centralité locale que l'on considère, à savoir si on considère la centralité comme le contrôle, l'indépendance ou l'activité.

En considérant une centralité locale de degré, le calcul de la centralité globale permet d'établir les points dominants, les centres d'intérêts, dans un réseau social, à savoir une activité concentrée autour de certaines ressources.

- Une mesure de la centralisation d'un réseau social, à partir des centralités locales d'intermédiarité, fournit un indice de la dépendance de l'efficacité de ce réseau par rapport à certains nœuds.

- Une mesure de la centralité globale d'un réseau, basée sur une centralité locale de proximité, permet de mesurer la performance de la communication dans ce réseau, notamment pour la circulation d'informations.

VII.3. Résistance d'un réseau social :

La mesure de la centralisation d'un réseau montre la dépendance d'un réseau par rapport à ses sommets. Cette dépendance peut également être mesurée par l'impact du retrait d'un sommet ou d'une arête sur la connectivité du réseau. En effet, le retrait d'un nœud ou d'une arête stratégique, par exemple un nœud ayant une forte centralité d'intermédiarité ou de proximité, peut augmenter la longueur du plus court chemin entre de nombreux autres nœuds voir scinder un réseau en deux ou plusieurs réseaux non reliés. Cette mesure s'effectue sur deux types de retraits possibles, des retraits aléatoires et des retraits ciblés. En général, les structures des réseaux sociaux sont assez résistantes à des retraits aléatoires de sommets ou d'arêtes alors qu'un retrait ciblé peut affecter sérieusement ces structures.

Par exemple, le retrait d'un pont entre deux groupes de sommets fortement connectés réduit considérablement voire coupe la communication entre ces deux groupes **[31]**.

VII.4. Détection de communautés

La détection de communautés permet, entre autres, de détecter les communautés non connectées. En connaissant les groupes fortement connectés, on peut aussi facilement déduire les sommets les plus intermédiaires. En plus de son lien étroit avec les notions précédemment mentionnées et tout particulièrement la centralité d'intermédiarité, la détection de communauté suscite d'autres intérêts. En effet dans un réseau social, la détection des communautés permet de déterminer la répartition des acteurs et des activités.

VIII. Représentation sémantique d'un réseau social

Avec le caractère toujours plus participatif du web, le paysage de la toile est désormais le produit de ses utilisateurs, devenus une des ressources majeures du web. En réponse à ce phénomène social, la communauté du web sémantique propose des modèles ontologiques pour représenter et exploiter les profils des utilisateurs, leurs usages et leur réseau social.

VIII.1. Modèles ontologiques :

L'initiative la plus célèbre et la plus adoptée est l'ontologie FOAF, Friend Of A Friend (ami d’un ami). Cette ontologie décrit "les personnes, les liens entre elles et ce qu'elles créent et font".

Tout d'abord un large ensemble de propriétés représentent la plupart des concepts nécessaires à la description d'un profil.

Exemple :

"le nom de famille", "omar" et "intérêt" permettent respectivement de définir le nom de famille, le prénom et un intérêt d'une personne. Ensuite la propriété "connaissance" est utilisée pour connecter les profils entre eux et ainsi former le réseau social des profils FOAF. Enfin FOAF modélise les usages des utilisateurs avec des

classes pour représenter les ressources manipulées (OnlineAccount, Document, Group…) et des propriétés pour les interactions des utilisateurs avec ces ressources (holdsOnlineAccount, weblog, member…).

VIII.2. Le social Tagging

Le social tagging consiste à partager des ressources et à les classifier avec des annotations sous forme de tags. Le fruit du social tagging est une classification de ressources librement établie par les utilisateurs, appelée folksonomie. L'adoption massive de cette pratique par les utilisateurs du web2.0 et la classification proposée par les folksonomies ont amené la communauté du web sémantique à s'intéresser de près à ces usages. Ainsi **[5]** pose les bases d'une ontologie décrivant les concepts essentiels d'une folksonomie. Il définit tout particulièrement le noyau d'une folksonomie, à savoir l'action de "tagging" composée d'une ressource, d'un tag et d'un utilisateur.

L'ensemble des tags manipulés par une personne ou un groupe de personnes est appelé un nuage de tags. Le nuage de tags est l'une des alternatives pour naviguer au sein des ressources d'une folksonomie.

L'ontologie SCOT **[32]** s'intéresse de près à ces nuages de tags et commence à s'imposer comme moyen de "représenter la structure et la sémantique des données du social tagging afin de les partager et de les réutiliser".

VIII.3. Représentation sémantique de personnes et d’usages

Dans la représentation sémantique des personnes et des usages, il est important de mentionner les microformats. Cette initiative est importante dans la marche en avant vers un web sémantique qui doit passer par une sémantique légère avant d'atteindre le but attendu par la communauté.

Le principe des microformats est d'utiliser les attributs de HTML de manière consensuelle dans l'optique d'ajouter de la sémantique embarquée dans un document XHTML. Les règles mises en place permettent de s'abstenir de l'usage d'une ontologie et de mettre en place un mécanisme de sémantique légère, sans règles d'inférence.

On retrouve ainsi un ensemble de microformats permettant de décrire des personnes, des ressources et des réseaux sociaux.

Exemples :

- Le microformat **hCard** pour représenter une carte de visite (nom, couriel, adresse, etc.).

- hResume pour la publication de CV et "XFN" (XTML Friends Network) pour décrire un réseau de connaissances sont des microformats qui permettent de représenter les profils des personnes.

De nombreux microformats sont destinés à la définition des ressources et usages du web: "hAtom" est utilisé pour la description des weblogs, "hCalendar" pour les évènements, "xfolk" pour les folksonomies, "votelink" pour les votes, "hReview" pour les revues sur les produits, "XMDP" pour les métadonnées d'une page, "adr" pour les adresses et "geo" pour la géo-localisation.

Des micros formats sont aussi disponibles pour définir la nature d'un lien hypertext en utilisant l'attribut "rel" de la balise <a>:

- rel="tag" pour les tags.
- rel="enclosure" pour les fichiers attachés.
- rel="nofollow" pour les liens à ne pas prendre en compte pour les algorithmes d'indexation.
- rel="directory" pour les liens vers un répertoire.
- rel="licence" pour les licences.

- rel="home" pour désigner une page d'accueil.

Grâce à leur facilité d'intégration, ces microformats sont largement utilisés notamment dans l'optique de la portabilité des données mais aussi pour une exploitation directe des informations (import d'une carte de visite dans son répertoire, ajout d'un évènement dans son agenda, visualisation sur une carte d'un lieu, etc.).

IX. Conclusion :

Dans ce chapitre la notion de réseau social a été défini, la structure, sémantique et ses représentation aussi bien graphiques que matricielles ont été abordées .Quelques indicateurs propres aux réseaux sociaux et a leurs plus amples compréhension ont été définis en s’appuyant sur le fait que le web a effectué un pas supplémentaire dans la socialisation en fournissant toujours plus d’interactions entre les individus connectés à Internet.

Notre objectif étant de concevoir et développer des techniques permettant de mettre en correspondances des communautés dans un réseau social une étape nécessaire à la création de cette entreprise s’impose, il s’agit de la conception.

I. Introduction

Après avoir présenté dans les chapitres précédents les concepts qui vont nous orienter afin de concevoir un réseau social permettant le partage de ressources, Il nous faut décrire dans ce chapitre le processus de développement de notre système.

Pour la description et la modélisation du système, une approche orientés objet a été choisi du faite que l'implémentation de celui-ci a été réalisé dans un langage de programmation objet. L'illustration de la modélisation s'est faite à l''aide de la représentation UML.

II. Présentation du langage UML

UML (Unifed Modeling Langage) est né au milieu des années 90, par la consolidation de quelques méthodes objets : OMT de James Rumbaugh, BOOCH de Gardy Booch et OOSE de Ivar Jacobson.

Comme son nom l'indique, UML n'est pas une méthode, mais un langage de modélisation. Il présente de nombreux avantages :

- Langage très expressif.
- Langage couvrant toutes les étapes nécessaires au développement d'un système.
- Simple à comprendre et à utiliser.
- Applicable à une large gamme de problème.
- Cohérent.
- Intuitif.

III. Analyse du système

Cette phase s'appuie sur les spécifications et vise à comprendre la problématique posée à laquelle le système à concevoir sera dédié. Elle est indépendante de toute

considération technique, informatique ou tout aspect de réalisation. A ce niveau le concepteur ne se préoccupe point de la manière dont il doit réaliser le système mais ce que doit faire ce dernier.

Pour cela, les 3 aspects suivants s'avère être nécessaire à étudier :

- Aspect fonctionnel (transformation des données).
- Aspect statique (modèle objet).
- Aspect dynamique (modèle dynamique).

III.1. Le modèle fonctionnel

Cet aspect peut être décrit par un diagramme de cas d'utilisation proposé par UML.

Pour ce faire, il faut tout d'abord spécifier les besoins, en commençant par la définition des principaux acteurs qui interagissent avec le système, puis comment est-ce que le système doit être utilisé ,en déterminant les cas d'utilisation possibles avec une description de chaque cas.

III.1.1 La spécification des besoins

Notre réseau social interagit avec des acteurs inscrites et appartenant à une communauté.

- **Détermination des acteurs de notre système :** les principaux acteurs recensés et qui interagissent dans le réseau social sont :

  - L'utilisateur : nous appelons utilisateur toute personne accédant, se connectant ou partageant des ressources sur le réseau social.

- **Détermination des opérations réalisées par les acteurs :** Les opérations que chaque acteur peut réaliser sont :

- L’utilisateur :

  - S’inscrire (s’enregistrer).
  - Accéder à l’actualité d’une communauté.
  - Effectuer une rechercher sur ses contact.
  - Tagguer un contact sur une ressource.
  - Partager une ressource ou une connaissance.

III.1.2. Spécification des scénarios :

Un scénario décrit la façon dont le système doit être utilisé ; en d’autre terme l’exécution pas à pas des différents cas d’utilisation. Dans le tableau ci-dessous, nous décrivons l’ensemble des scénarios relatifs à chaque acteur :

Acteurs	Taches	Scénarios
Utilisateur	Accéder à la page d'accueil.	1. Ouvrir le navigateur Internet. 2. Saisir l'URL du site.
	S'enregistrer	3. Saisir un identifiant. 4. Saisir un mot de passe. 5. Saisir une adresse E-mail. 6. Valider l'opération.
	Ajouter un contact	7. Sélectionner un contact à ajouter dans la liste.
	Partager une ressource	8. Partager avec un contact une ressource sélectionnée.
	Tagguer un contact sur une ressource	9. Sélectionner le chemin de la ressource. 10. Sélectionner le contact. 11. Valider l'opération.

	Effectuer une recherche sur un signet.	12. Introduire le signet. 13. Introduire le contact. 14. Lancer la recherche.

Tableau III.1.Tableau représentant les scénarios.

III.1.3. Spécification des cas d'utilisation :

- Le cas d'utilisation « s'enregistrer ».

  Use case: s'enregistrer.

  Scenarios: 3,4,5,6.

  Role: L'utilisateur.

  Description:

  1. L'utilisateur ouvre la fenêtre du navigateur (Mozzila Fire Fox, Internet explorer, Netscape,….).
  2. L'utilisateur saisit l'adresse URL relatif au site de partage de connaissance.
  3. Le navigateur lui affiche une page web.
  4. L'utilisateur s'enregistre en spécifiant :

     - Un identifiant
     - Un mot de passe
     - Une adresse E-mail.

  5. L'utilisateur valide l'opération.

- Le cas d'utilisation « partager une ressource ».

  Use case: partager une ressource.

  Scenarios: 8.

  Role: L'utilisateur.

  Description:

  1. L'utilisateur sélectionne une ressource à partager.
  2. L'utilisateur inscrit le chemin physique ou l'adresse (URL de cette ressource).
  3. L'utilisateur valide l'opération.

- Le cas d'utilisation « tagguer une ressource ».

  Use case: partager une ressource.

  Scenarios: 9, 10, 11.

  Role: L'utilisateur.

  Description:

  1. L'utilisateur sélectionne le chemin physique ou l'URL d'une ressource.
  2. L'utilisateur sélectionne un contact parmi sa liste de ses contacts.
  3. L'utilisateur inscrit des descriptions sur la ressource.
  4. L'utilisateur valide l'opération.

- Le cas d'utilisation «effectuer une recherche sur un signet ».

  Use case: rechercher un signet.

  Scenarios: 12, 13, 14.

  Role: L'utilisateur.

  Description:

  1. L'utilisateur introduit le nom du signet.
  2. L'utilisateur introduit éventuellement le nom du contact correspondant au signet.
  3. L'utilisateur valide l'opération.

II.1.4. Diagramme des cas d'utilisations :

Un cas d'utilisation (use case) modélise une interaction entre le système informatique à développer et un utilisateur ou acteur interagissant avec le système.

Plus précisément, un cas d'utilisation décrit une séquence d'actions réalisées par le système qui produit un résultat observable pour un acteur.

La présentation d'un cas d'utilisation met en jeu trois concepts :

- Les acteurs.
- Les scénarios.
- L'interaction entre l'acteur et le cas d'utilisation.

Les interactions entre les cas d'utilisations peuvent être décrites par trois relations.

1. **Relation d'inclusion (includ) :** Elle est employée quand deux cas d'utilisation ont en commun une même fonctionnalité et que l'on souhaite factoriser celle-ci en créant un sous cas, ou cas intermédiaire, afin de marquer les différences d'utilisation.
2. **Relation d'extension « étend » :** Schématiquement, nous dirons qu'il a extension d'un cas d'utilisation quand un cas est globalement similaire à un autre, tout en effectuant un peu plus de travail (voire un travail plus spécifique).

   Cette notion –à utiliser avec discernement-permet d'identifier des cas particulier (comme des procédures à suivre en cas d'incident) dés le début ou lorsque l'attitude face à un utilisateur spécifique du systéme doit être spécialisée ou adaptée. Il s'agit grosso modo d'une variation du cas d'utilisation normale.
3. **Relation de généralisation :** Une relation de généralisation de cas d'utilisation peut être définie conformément au principe de la spécialisation défini pour les classes.

   La généralisation est la relation entre une classe et deux autre classe ou plus partagent un sous ensemble commun d'attributs et/ou d'opérations.

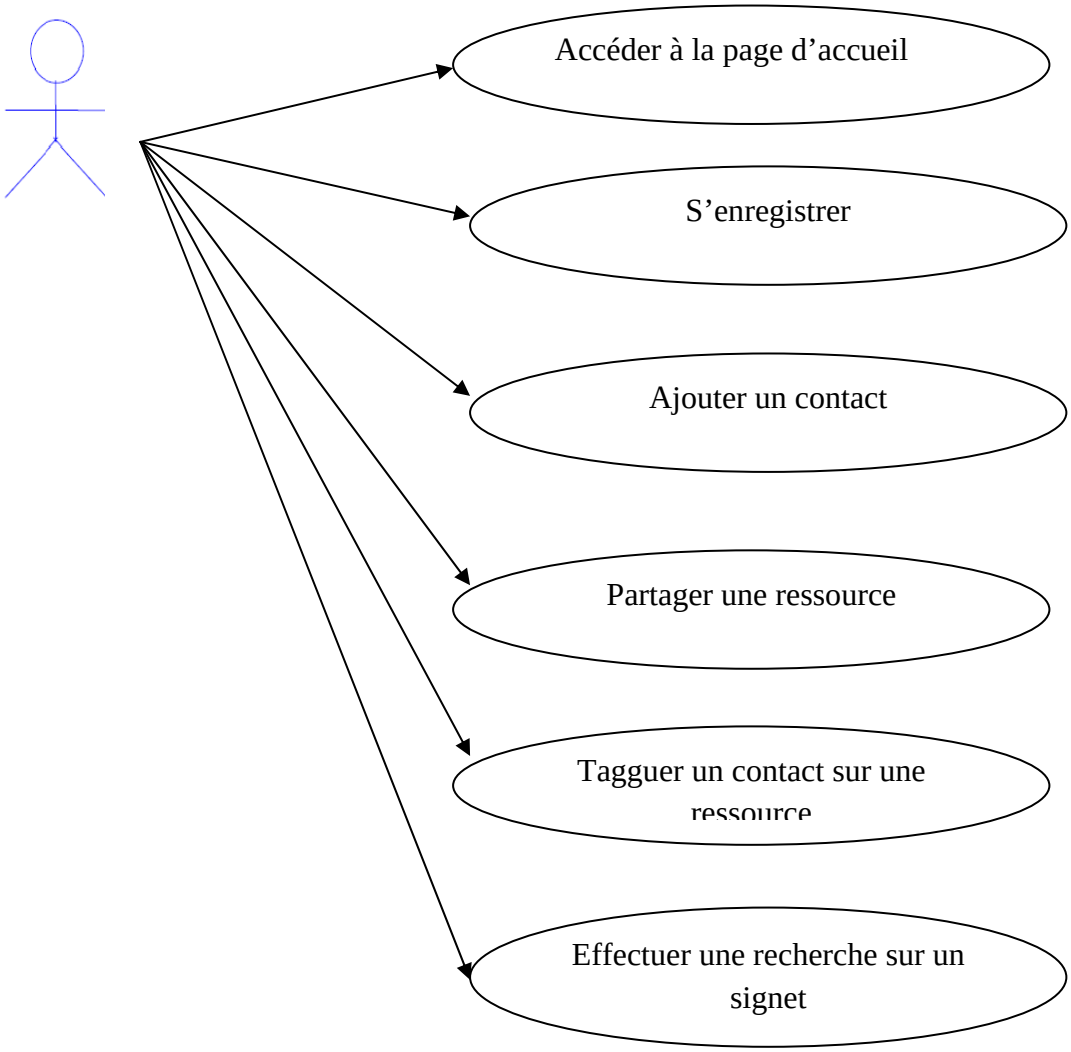


Figure III.1 : Diagramme des cas d'utilisations (uses case).

III.2. Le modèle dynamique :

La modélisation dynamique permet de modéliser le flot d'un objet lorsqu'il transite d'un état à un autre. Il est écrit par les diagrammes de séquence et d'activité d'UML.

III.2.1. Le diagramme de séquence :

Il illustre la dynamique d'enchainement des traitements d'une application effectuée par le système. Ces traitements sont ordonnés dans le temps traduisant ainsi la chronologie des événements entre les différents objets du système. Ce type de diagrammes a le principal intérêt d'illustrer les cas d'utilisation.

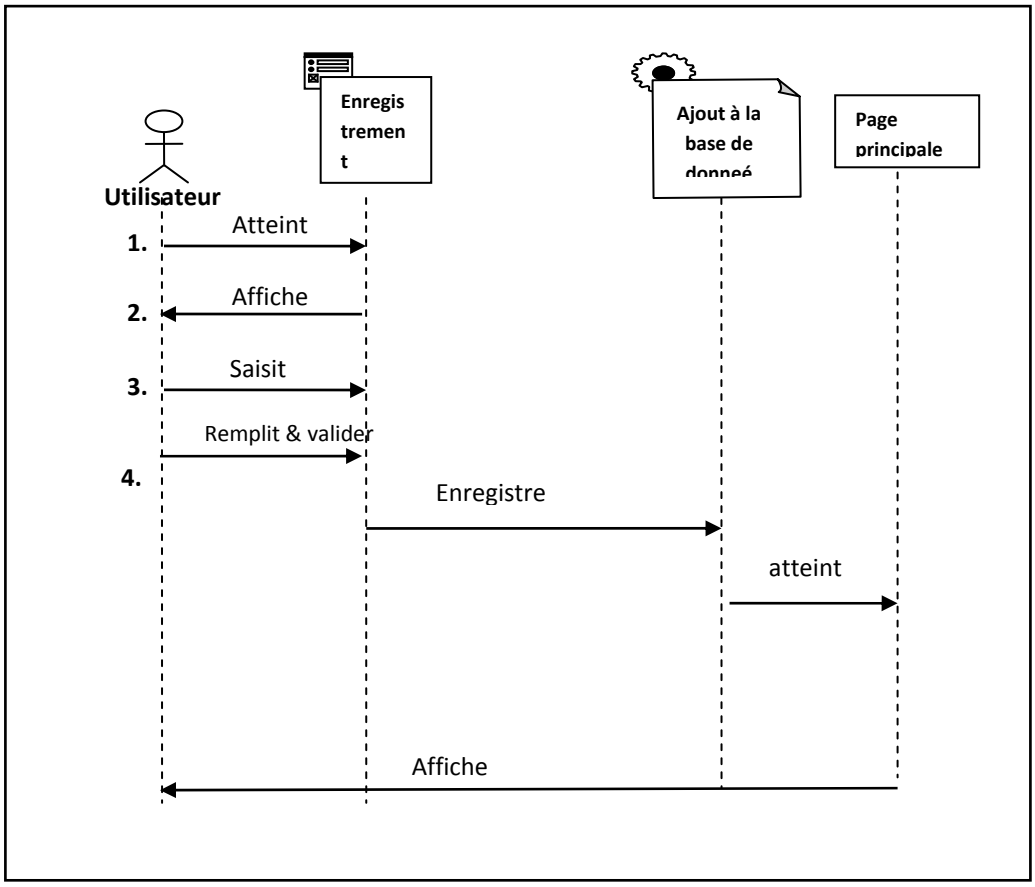


Figure III.2. Diagramme de séquence de réalisation de cas d'utilisation « S'enregistrer »

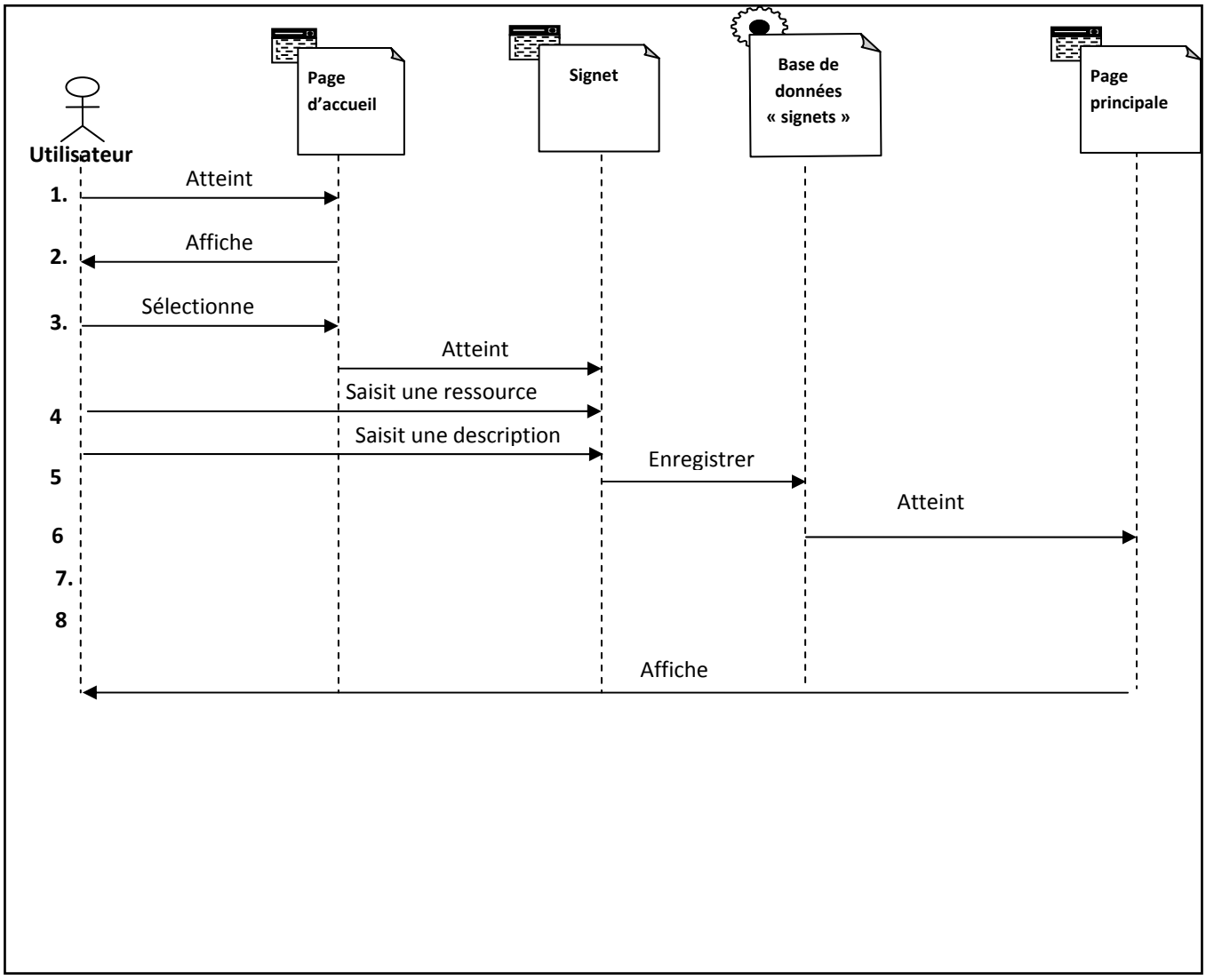


Figure III.3. Diagramme de séquence de réalisation de cas d'utilisation « Partager un signet »

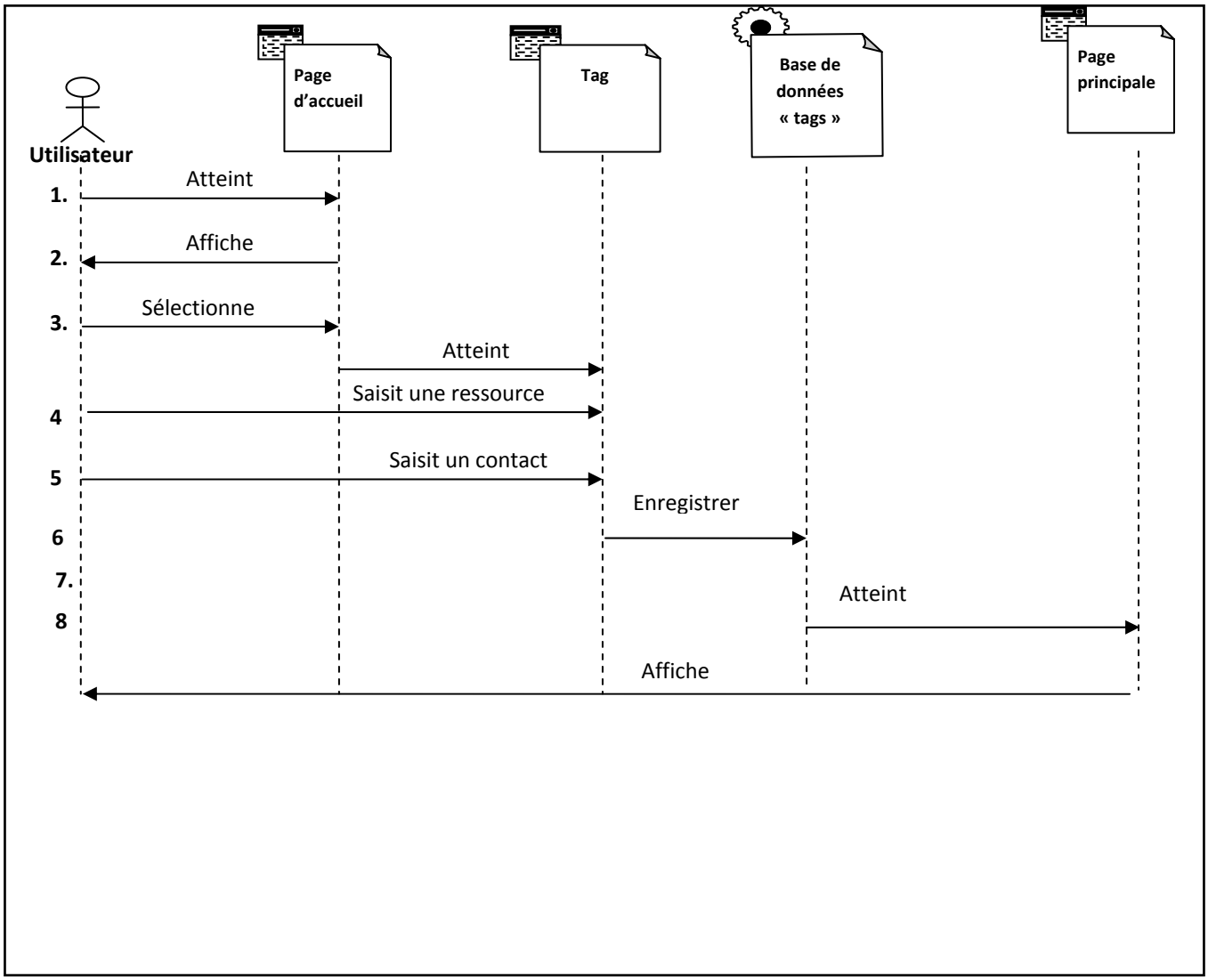


Figure III.4. Diagramme de séquence de réalisation de cas d'utilisation «Tagguer un contact sur une ressource»

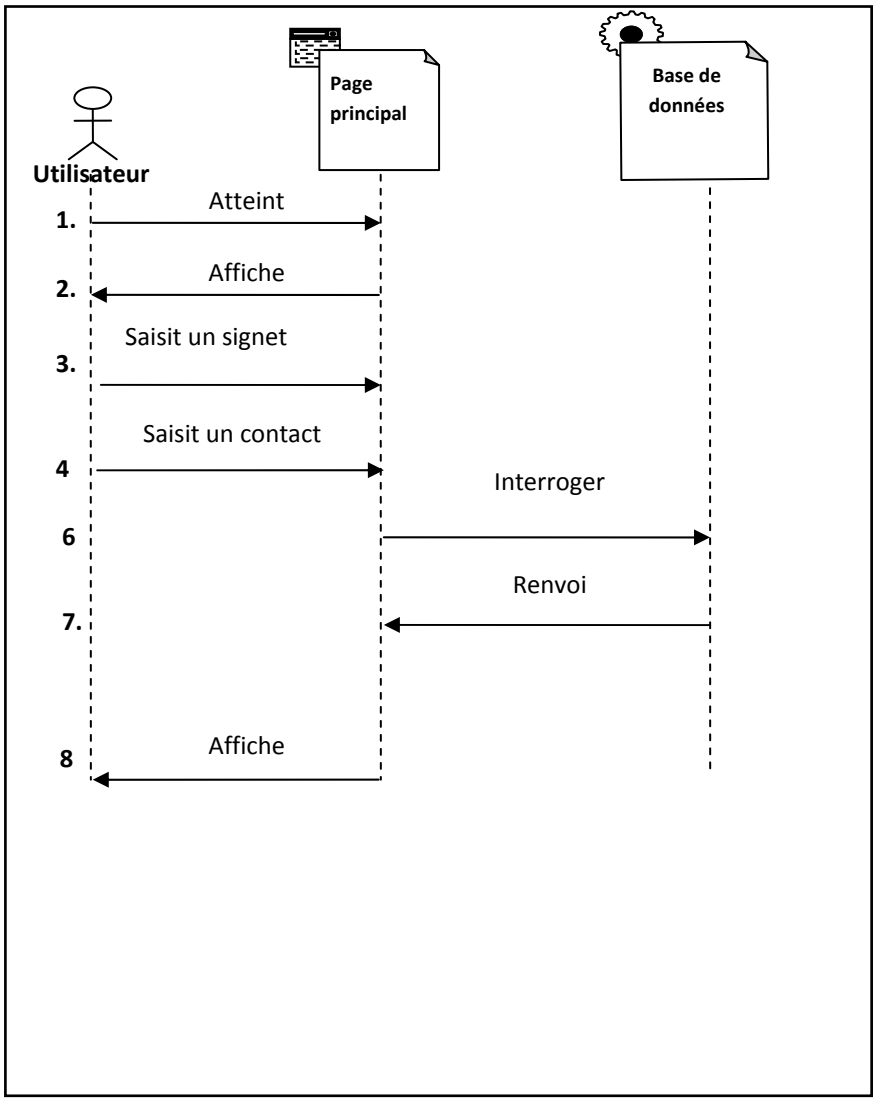


Figure III.5.Diagramme de séquence de réalisation de cas d'utilisation «Rechercher un signet»

III.3. Le modèle statique :

Cet aspect peut être décrit par le diagramme de classe (figure III.6) et le diagramme d'objet d'UML. Du au faite que ce dernier n'est qu'une dérivation du premier on va se limiter à modéliser notre système par le diagramme de classe.

III.3.1. Le diagramme de classe :

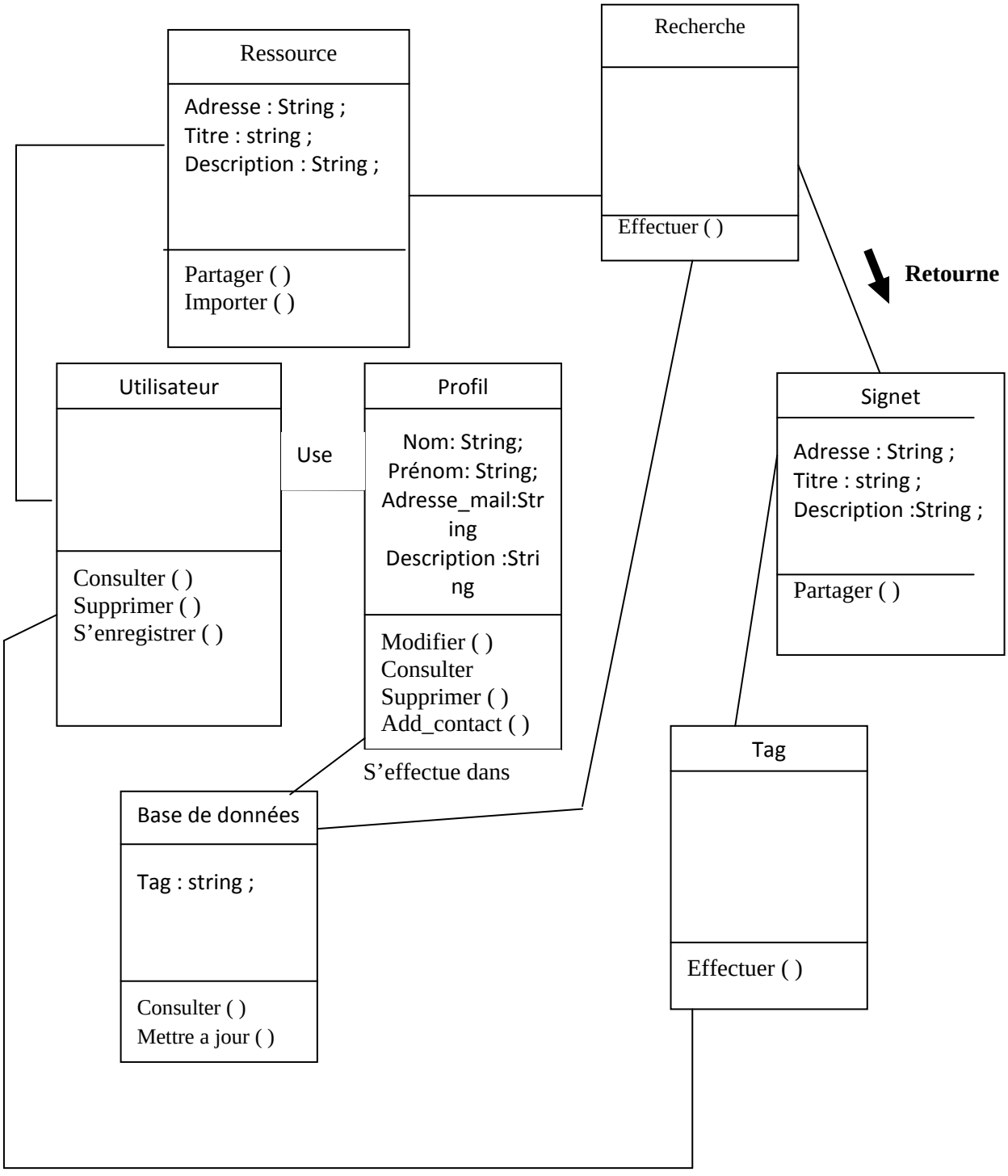


Figure III.6. Diagramme des classes

IV. Modélisation du profil utilisateur

La modélisation du profil utilisateur constitue un élément essentiel dans le développement du système dont l'efficacité dépend fortement de la précision des profils, et pour cela il faut cibler les paramètres qui caractérisent au mieux un utilisateur en quête d'informations pour les modéliser dans son profil qui est le composant principale de notre système. Nous prendrons en considération des objectifs pour modéliser le profil utilisateur parmi eux :

1. Proposer une description afin de représenter au mieux les besoins Informationnels de l'utilisateur.

2. Catégoriser et hiérarchiser les mots- clés qui décrivent ces besoins.

3. Avoir une description suffisamment flexible pour pouvoir lui apporter au besoin, les Mise à jour et améliorations nécessaires.

4. Intégrer le profil dans un système de recherche d'information pour mesurer l'impact de sa modélisation sur les performances du système.

La figure III.7 illustre une description sémantique du profil utilisateur.

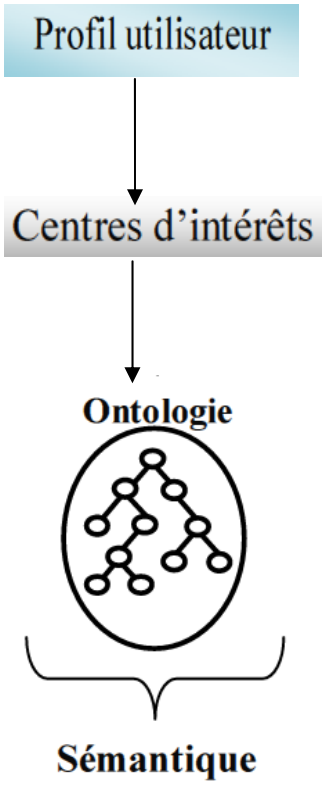


Figure III.7. Description sémantique du profil utilisateur.

IV.1. Centres d'intérêts

Le centre d'intérêt exprime le domaine qui intéresse l'utilisateur ou son périmètre d'exploration, et permet aussi une présélection virtuelle. Par conséquent toute requête émise par l'utilisateur sera enrichi avec les mots clés ou les prédicats des requêtes définissant le centre d'intérêt. Dans notre travail, nous avons proposé une ontologie pour les centres d'intérêts, le but de cette ontologie est de décrire de manière sémantique les centres d'intérêts de l'utilisateur et décrire (enrichir) la requête par les même concepts pour faciliter ensuite l'appariement.

Durant la recherche l'utilisateur peut sélectionner ces centres d'intérêts à partir de la liste des centres de l'ontologie ou ajouter manuellement des centres d'intérêts non trouvés dans l'ontologie ces derniers sont transmis vers le module d'enrichissement de l'ontologie où un spécialiste (expert de domaine) les affectés.

Pour la construction de l'ontologie nous avons utilisé l'approche proposée par l'université de **Stanford [33]** parce qu'elle comporte des étapes claires, simples et faciles à comprendre. Ajouter à cela, l'outil avec lequel nous allons construire l'ontologie en l'occurrence « protégé 4.0.2» est développé par la même université.

IV.2. Construction de l'ontologie

Nous avons procéder a la construction de cette ontologie, par le recensement et l'analyse des centres d'intérêts établie comme étant une description de son profil. Ceci permet de connaitre la catégorie des utilisateurs ciblés.

Etape 1 : Déterminer le domaine et la portée de l'ontologie.

- Le domaine que va couvrir l'ontologie est le domaine du e-Learning .
- Le but de l'utilisation de l'ontologie est l'enrichissement sémantique des requêtes utilisateur en utilisant la hiérarchie des concepts.
- L'ontologie doit répondre aux questions correspond à la hiérarchie des concepts.

L'ontologie sera utilisée par le système qui va la parcourir pour trouver les concepts les plus similaires à un concept d'une requête.

- L'ontologie sera maintenue par un expert de domaine informatique.

Etape 2 : Envisager une éventuelle réutilisation des ontologies existantes.

Il est toujours utile de prendre en considération ce que d'autres personnes ont fait et d'examiner si nous pouvons élargir des sources existantes et les affiner pour répondre aux besoins de notre domaine ou de notre tâche particulière. Mais dans notre situation les ontologies existantes sont très vastes et beaucoup généralisées par contre nous avons besoin d'une ontologie spécialisée pour le domaine du e-Learning.

Etape 3 : Enumérer les concepts importants dans l'ontologie.

Nous avons collecté les concepts de domaine après une étude sur le domaine du e-Learning, ces concepts sont généralement fréquents dans un échantillon des utilisateurs: dans leurs travaux de recherche, les documents en relation avec leur domaine et dans leurs requêtes de recherche.

Voici quelque concepts: système d'information, analyse, conception, développement, management du changement, décision, connaissance, organisation, sécurité, système interactif d'aide à la décision, intégration des si, intégration des données, intégration des application, web service, service web sémantique, …etc.

Etape 4 : Définir les classes et la hiérarchie des classes.

Comme nous avons cité dans l'étape "1" notre ontologie sera utilisée pour enrichir la requête par des concepts voisins de concepts abordés dans la requête, alors les concepts collectés dans l'étape précédente sont tous des classes et nous n'avons pas besoin ces propriétés et ces instances donc les étapes :

Étape 5 : Définir les propriétés des classes – attributs.

Étape 6 : Définir les facettes des attributs.

Étape 7 : Créer les instances.

IV.3. Présentation de l'indexation

L'indexation est une étape très importante dans le processus de RI. Elle consiste à déterminer et à extraire les concepts représentatifs du contenu d'un document. La qualité de la recherche dépend en grande partie de la qualité de l'indexation. Le résultat de l'indexation constitue, ce que l'on nomme le ***descripteur*** du document. Ce dernier est souvent une liste de concepts ou groupe de concepts significatifs pour l'unité textuelle correspondante, généralement assortis de poids représentant leur degré de représentativité du contenu sémantique de l'unité qu'ils décrivent. Les descripteurs des

documents (mots, groupe de mots) sont rangés dans un catalogue appelée dictionnaire constituant le ***langage d'indexation***.

IV.4. Module d'indexation

Le processus de l'indexation effectue le transfert de l'information contenue dans des documents vers un autre espace de représentation traitable par un système informatique **[34].**A partir d'une collection de documents, le processus d'indexation nous renvoie une liste des mots clés structurés comme le montre la **Figure III.8.** Ce résultat est le plus souvent utilisé pour effectuer des recherches d'informations. Mais il peut servir également pour comparer et classifier des documents, proposer des mots-clés, faire une synthèse automatique de documents, calculer des co-occurrentes de termes **[34],** etc.

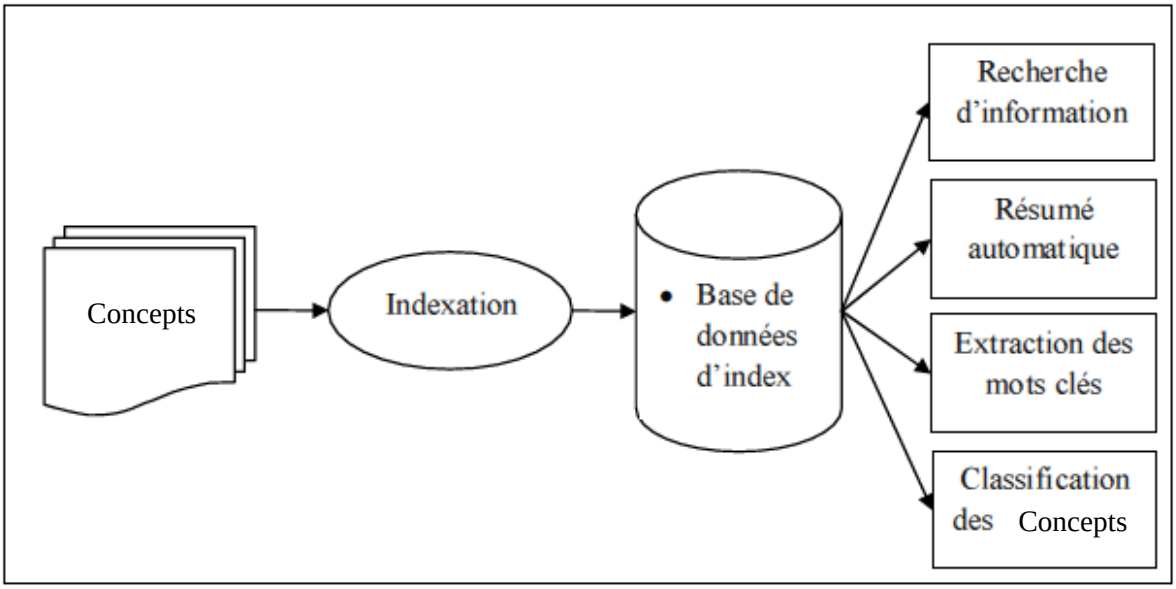


Figure III.8.Module d'indexation

V. Conclusion :

Durant l'étape d'analyse et de conception les besoins auxquels doit répondre le système ont été décrits, et analyser à l'aide des différents diagrammes proposés par le langage UML avec la représentation des centres d'intérêts du profil d'un utilisateur sous forme d'ontologie afin de l'utiliser dans le module de recherche en citant en détaille la construction de cet l'ontologie. Par la suite la conception du système, a permis d'exposer l'ensemble des fonctionnalités qu'on lui a défini, ce qui permettra de répondre à tous ces besoins, le chapitre suivant sera dédier à la réalisation du système en explicitant les plates formes utilisées ainsi que les moyens mise en œuvre pour sa conception.

I. Introduction

La réalisation est la partie finale du développement logiciel, elle aborde les détails de codage en employant un langage de programmation.

Dans le chapitre précédent, nous l'étape de conception du réseau social fut détaillée ainsi que les outils utilisés pour le parage de connaissance et de ressources. Les résultats de cette étape seront exploités dans la présentation de l'environnement de développement puis nous présentons quelques interfaces du réseau social.

Dans ce chapitre nous présenterons aussi les architectures logiques et physiques adoptées pour le déploiement de l'application ainsi que certains détails techniques de l'implémentation.

II. Architecture technique

L'architecture technique (communément appelée architecture physique) décrit l'ensemble des composants matériels supportant une application.

Ainsi l'architecture logique du système adoptée est de type «Architecture 2-Tier».

- la **présentation** des données : correspondant à l'affichage, la restitution sur le poste de travail, le dialogue avec l'utilisateur ;
- le **traitement** métier des données : correspondant à la mise en œuvre de l'ensemble des règles de gestion et de la logique applicative ;
- et enfin l'**accès aux données** persistantes : correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive.

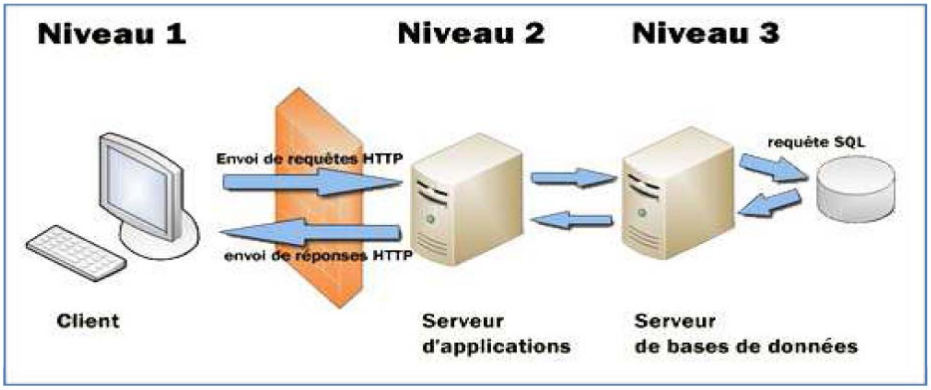


Figure IV.1 Architecture en 3-tier

Ce modèle d'architecture 3-tiers a pour objectif de répondre aux préoccupations suivantes :

- Allègement du poste de travail client (notamment vis-à-vis des architectures classiques client-serveur) ;
- Prise en compte de l'hétérogénéité des plates-formes (serveurs, clients, langages, etc.) ;
- Amélioration de la sécurité des données.
- Meilleure répartition de la charge entre différents serveurs d'application.

II.1 Déploiement du système

Un diagramme de déploiement décrit la disposition physique des ressources matérielles qui composent le système et montre la répartition des composants sur ces matériels. Chaque ressource étant matérialisée par un nœud, le diagramme de déploiement précise comment les composants sont répartis sur les nœuds et quelles sont les connexions entre les composants ou les nœuds.

Voici dans ce qui suit, la description des différents nœuds ainsi que leurs composants:

II.1.1 Le poste client

Il s'agit des postes de travail des différents utilisateurs et visiteurs du portail. Ce sont de simple PC devant être équipés d'un navigateur web (FireFox, Opéra, Internet Explorer).

II.1.2 Le serveur de base de données

Cette machine hébergera la base de données sur laquelle tournera notre application. Le SGBD utilisé sur le serveur sera MySql qui est le plus adéquat et le plus utilisé en matière de portails web pour sa rapidité et ça simplicité.

II.1.3 Serveur

Ce serveur héberge notre serveur d'application(Apache) ainsi que le composant du site.

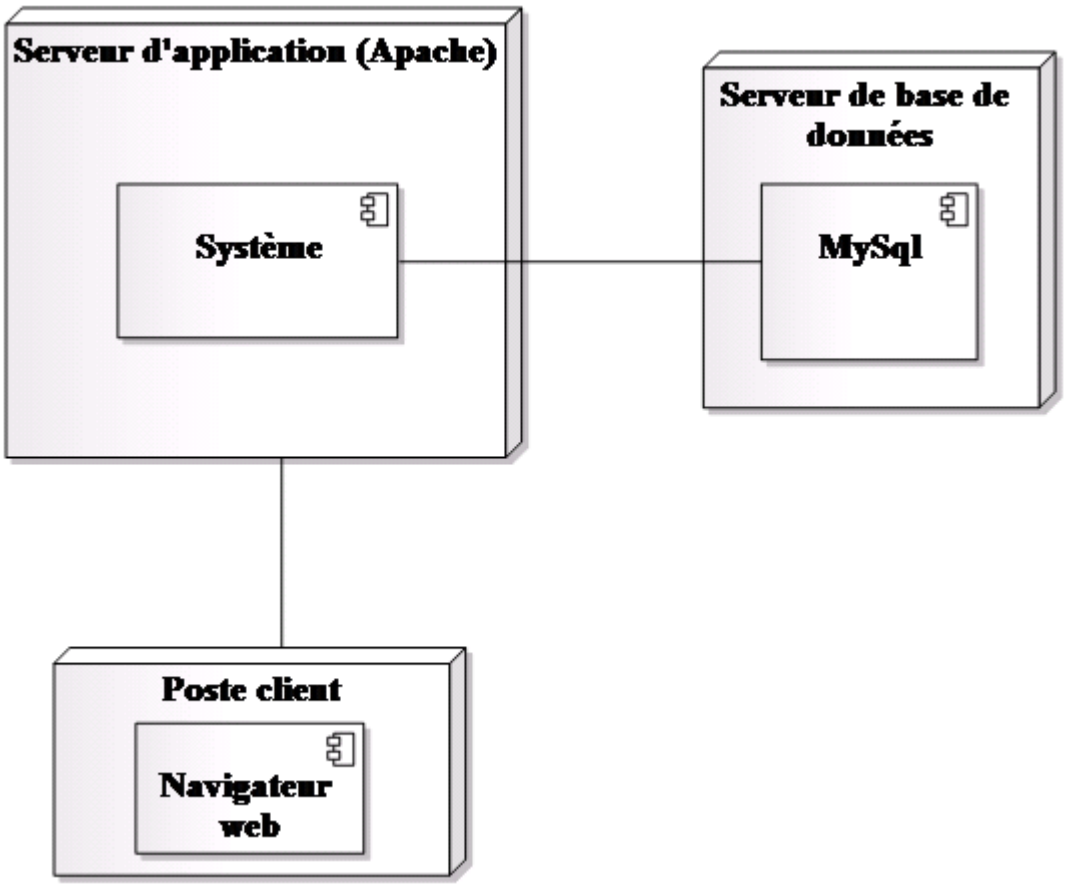


Figure IV.2 Diagramme de déploiement de la plate forme

III. L'environnement de développement

Dans cette section nous allons présenter les différents langages et outils utilisés pour le développement de notre portail web. Ainsi nous commencerons par les aspects du coté serveur puis coté client et nous finirons par des API que nous avons intégré.

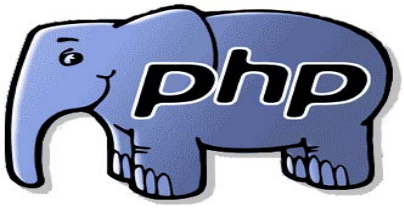
III.1Coté serveur

Apache : Apache HTTP Server est un logiciel libre de serveur HTTP produit par l' « Apache Software Foundation ».

Symbole d'Apache

Il s'agit d'un serveur sécurisé, performant et évolutif donnant accès aux services HTTP en accord avec les derniers standards de ce protocole. Ainsi ses performances, sa robustesse, son utilisation aisée, et sa licence en font le serveur web le plus populaire au monde depuis mars 1997.

PHP : (Hypertext Preprocessor) est un langage de scripts libre principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale, en exécutant les programmes en ligne de commande. Ainsi les principaux atouts qui nous ont poussés à choisir PHP sont :



Mascotte de PHP

- PHP est un langage de scripts. Il est interprété, par conséquent il ne nécessite pas d'être compilé pour obtenir un objet, un exécutable avant d'être utilisable.
- Un de ses grands avantages est l'intégration dans la même page du code HTML « brut » et du code PHP. Il s'imbrique dans le code HTML en étant délimité comme tel. Ainsi, les scripts PHP ne nécessitent pas de répertoires spéciaux.
- PHP est un module supporté par le serveur web Apache, il est donc développé pour être facilement utilisable via ce serveur (il fonctionne évidemment avec d'autres serveurs web comme IPlanet, IIS...).
- PHP reconnait l'essentiel des protocoles et formats disponibles sur Internet et intranet : TCP, HTTP, SMTP, LDAP, IMAP, POP, SSl, Soap, XML, PDF.

MySql : C'est un système de gestion de base de données (SGBD).

Selon le type d'application, sa licence est libre ou propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle et Microsoft SQL Server. Ses principaux avantages sont sa:

- Portabilité (UNIX, Windows, MacOs X).
- Haute vitesse d'exécution.
- Facilité d'installation et d'administration (par rapport aux autres SGBD).

Pour conclure, le couple PHP/MySQL est très utilisé par les sites Web et proposé par la majorité des hébergeurs. Apache étant le plus souvent utilisé conjointement avec PHP et MySQL.

Logo MySQL

L'API JSP fait partie de J2EE (Java 2 Entreprise Edition), elle donne aux développeurs les moyens de développer des applications Web de façon simple et puissante.

JSP permet de séparer la logique programmatique (le code java) de la présentation (les balises HTML.

Exécution d'une JSP Un conteneur de JSP (JSP container) fournit les services réseaux par lesquels les requêtes et les réponses sont émises, il décode également les requêtes et formate les réponses dans le format approprié. Tous les conteneurs doivent supporter le protocole HTTP.

III.2 Coté Client

JavaScript : JavaScript est un langage orienté objet de programmation de scripts principalement utilisé dans les pages web interactives.

JavaScript sert à contrôler les données saisies dans des formulaires HTML, ou à interagir avec le document HTML via l'interface DOM (Document Object Model), fournie par le navigateur (on parle alors de HTML dynamique ou DHTML).

AJAX (Asynchronous Javascript And XML): « Ajax n'est pas une technologie, il s'agit de plusieurs technologies se développant chacune de leur côté et combinées pour donner des résultats aussi nouveaux que puissants. Ajax comporte :

- Une présentation fondée sur les standards XHTML et CSS ;
- Un affichage dynamique et interactif grâce à DOM (*Document Object Model*) ;
- Un système d'échange et de manipulation de données utilisant XML et XSLT mais aussi JSON ;
- Un mécanisme de récupération de données asynchrones utilisant XMLHttpRequest ;
- JavaScript pour lier le tout. »

JQuery : c'est une bibliothèque JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant AJAX) et HTML, et a pour but de simplifier des commandes communes de JavaScript.

Protégé 4.0.2: (téléchargé à l'adresse http://protege.stanford.edu)

Plusieurs outils sont dédiés à la création d'ontologies, par exemple Ontoedit, OntoTerm, Oiled, WebOnto, OntoSaurus ou encore Ontolingua. Cependant, ces logiciels utilisent un formalisme spécifique et ne sont pas couplés à un système d'indexation et de recherche. Et aussi pour d'autre contraintes techniques nous avons trouvé que l'éditeur protégé 4.0.2 développé par l'université de Stanford est le mieux adapté pour développer notre ontologie.

Protégé 4.0.2 est un outil software intégré utilisé par les développeurs de système et les experts de domaine pour développer des systèmes à base de connaissances. Les applications développées par Protégé 4.0.2 sont utilisées pour résoudre des problèmes et prendre des décisions dans un domaine bien précis.

Pendant que les systèmes à base de données classiques séparent la définition des classes d'information et le stockage des instances de ces classes, Protégé 4.0.2 permet de travailler avec les classes et les instances en même temps.

Protégé 4.0.2 est actuellement utilisé dans la médicine clinique et les sciences biomédicales. Il peut être utilisé dans tout champ où les concepts peuvent être modélisés comme une hiérarchie de classes. La figure IV.2 donne l'interface générale de protégé 4.0.2.

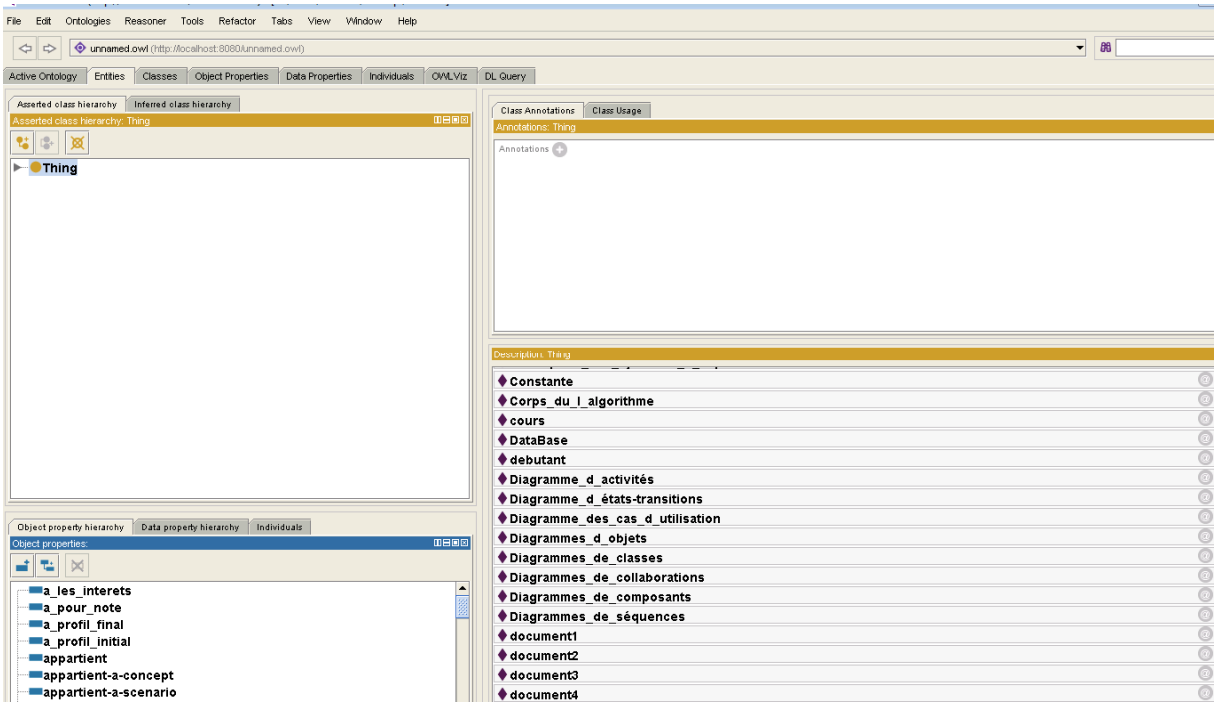


Figure IV.3 : Interface générale de protégé 4.0.2

Le schéma XML de notre ontologie :

Nous donnons ci-dessous le schéma XML qui décrit le formalisme et la structure de notre ontologie, l'ensemble des classes représentant les concepts de base, les slots associés à chaque classe ainsi que les relations entre les classes.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xs:schema elementFormDefault="qualified" targetNamespace="http://protege.stanford.edu/"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="file:///d:/base.xsd" />
  - <xs:complexType name="ConceptType">
    - <xs:complexContent>
      - <extension base="THING">
        - <xs:sequence>
            <xs:element fixed="STANDARD-CLASS" maxOccurs="0"
                minOccurs="0" name="METACLASS" type="xs:string" />
            <xs:element maxOccurs="unbounded" minOccurs="0"
                name="composer_de" type="Instance" />
            <xs:element minOccurs="0" name="contrôler" type="Instance" />
            <xs:element minOccurs="0" name="dépendre" type="Instance" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="Elte-cpt"
                type="Instance" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="gère"
                type="Instance" />
            <xs:element minOccurs="0" name="nom_cpt" type="xs:string" />
            <xs:element minOccurs="0" name="num_cpt" type="xs:integer" />
            <xs:element maxOccurs="unbounded" minOccurs="0"
                name="Précéder_par" type="Instance" />
            <xs:element minOccurs="0" name="utiliser" type="Instance" />
            <xs:element maxOccurs="0" minOccurs="0" ref="SEPARATOR" />
          </xs:sequence>
        </extension>
      </xs:complexContent>
    </xs:complexType>
  - <xs:complexType name="EltcType">
    - <xs:complexContent>
      - <extension base="THING">
        - <xs:sequence>
            <xs:element fixed="STANDARD-CLASS" maxOccurs="0"
                minOccurs="0" name="METACLASS" type="xs:string" />
            <xs:element maxOccurs="unbounded" minOccurs="0"
                name="composer-de" type="Instance" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="dépend"
                type="Instance" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="est_la"
                type="Instance" />
            <xs:element minOccurs="0" name="faire" type="Instance" />
            <xs:element maxOccurs="unbounded" minOccurs="0"
                name="fait_appel" type="Instance" />
            <xs:element minOccurs="0" name="nom_eltc" type="xs:string" />
            <xs:element minOccurs="0" name="num_eltc" type="xs:integer" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="précède"
                type="Instance" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="sorte_de"
                type="Instance" />
            <xs:element maxOccurs="unbounded" minOccurs="0"
                name="sous_forme_de" type="Instance" />
```

```
                <xs:element maxOccurs="unbounded" minOccurs="0" name="utilise"
                    type="Instance" />
                <xs:element maxOccurs="0" minOccurs="0" ref="SEPARATOR" />
              </xs:sequence>
            </extension>
          </xs:complexContent>
        </xs:complexType>
      - <xs:complexType name="ScénarioType">
          - <xs:complexContent>
              - <extension base="THING">
                  - <xs:sequence>
                      <xs:element fixed="STANDARD-CLASS" maxOccurs="0"
                          minOccurs="0" name="METACLASS" type="xs:string" />
                      <xs:element maxOccurs="unbounded" minOccurs="0" name="Elte-sc"
                          type="Instance" />
                      <xs:element maxOccurs="unbounded" minOccurs="0" name="Eltp"
                          type="Instance" />
                      <xs:element minOccurs="0" name="nom-sc" type="xs:string" />
                      <xs:element minOccurs="0" name="num_sc" type="xs:integer" />
                      <xs:element maxOccurs="unbounded" minOccurs="0"
                          name="précéder_par" type="Instance" />
                      <xs:element maxOccurs="0" minOccurs="0" ref="SEPARATOR" />
                    </xs:sequence>
                </extension>
            </xs:complexContent>
        </xs:complexType>
      - <xs:complexType name="DidacticielType">
          - <xs:complexContent>
              - <extension base="THING">
                  - <xs:sequence>
                      <xs:element fixed="STANDARD-CLASS" maxOccurs="0"
                          minOccurs="0" name="METACLASS" type="xs:string" />
                      <xs:element maxOccurs="unbounded" minOccurs="0" name="concept"
                          type="Instance" />
                      <xs:element minOccurs="0" name="nom_didac" type="xs:string" />
                      <xs:element maxOccurs="unbounded" minOccurs="0" name="scénario"
                          type="Instance" />
                      <xs:element maxOccurs="0" minOccurs="0" ref="SEPARATOR" />
                    </xs:sequence>
                </extension>
            </xs:complexContent>
        </xs:complexType>
      - <xs:complexType name="projetOntologyType">
          - <xs:sequence>
              <xs:element maxOccurs="unbounded" minOccurs="0" name="Slot"
                  type="SlotType" />
              <xs:element maxOccurs="unbounded" minOccurs="0" name="SlotOverride"
                  type="Slot_Override_Type" />
              <xs:element maxOccurs="unbounded" minOccurs="0" name="Concept"
                  type="ConceptType" />
              <xs:element maxOccurs="unbounded" minOccurs="0" name="Didacticiel"
                  type="DidacticielType" />
              <xs:element maxOccurs="unbounded" minOccurs="0" name="Eltc"
                  type="EltcType" />
              <xs:element maxOccurs="unbounded" minOccurs="0" name="Scénario"
                  type="ScénarioType" />
            </xs:sequence>
        </xs:complexType>  <xs:element name="projetOntology" type="projetOntologyType" />
</xs:schema>
```

IV. Présentation des interfaces du Réseau social

Pour réaliser une application Web il faut tenir compte de la qualité de l'interface homme/machine et permettre une meilleure adéquation de l'application aux besoins des différents utilisateurs.

Dans ce qui suit nous quelques interfaces de l'application sont présentés ci-dessous :

IV.1 Page d'enregistrement

Cette page est nécessaire afin qu'un quelconque utilisateur puisse accéder au réseau.

L'utilisateur doit entrer :

- Son nom (au moins 5 caractères).
- Son mot de passe.
- Son adresse E-mail.
- Et doit répondre à la question anti spam.

Figure IV.4 Page d'enregistrement

IV.2 Partager une ressource

Après s'être enregistré l'utilisateur peut partager une ressource avec les membres inscrits du réseau.

Il lui suffit d'entrer :

- Une adresse (ou se situe la ressource).
- Un titre
- Une description de la ressource.

Notons que :

L'utilisateur peut aussi tagguez un contact sur une ressource qu'il juge être intéressante.

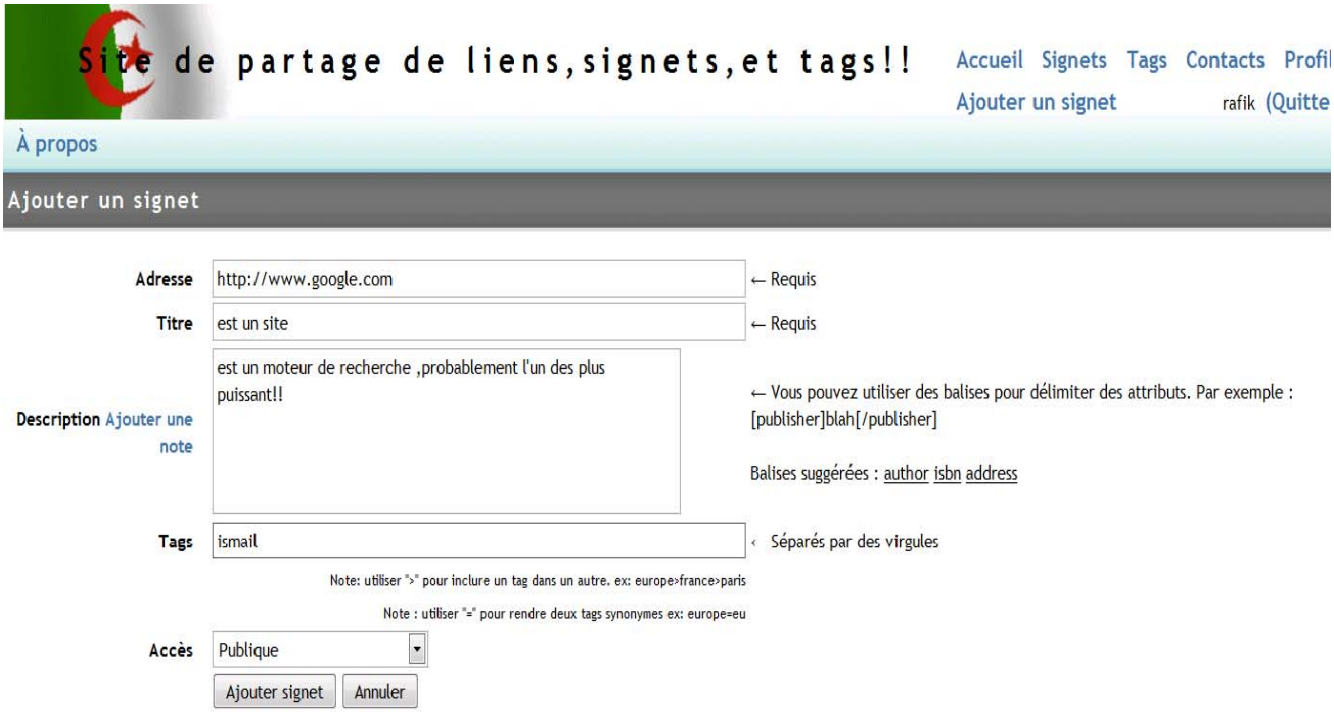


Figure IV.5. Page permettant de partager une ressource

Une fois les formulaires remplis la ressource partagée est visible parmi les utilisateurs connectés au réseau (figure IV.5).

Figure IV.6. Page montrant que la ressource a été partagée et est visible.

La ressource a été partagée sur le réseau.

Les autres utilisateurs peuvent :

- Voter pour ou contre.
- Tagguer un autre utilisateur sur cette ressource.

Notons que :

Les ressources partagées peuvent être triés selon :

- La date d'ajout
- Le titre (ordre alphabétique)
- Le vote des autres utilisateurs (les plus populaires)

III.3 Editer Son profil

L'utilisateur peut éditer don profil et peut ajouter en plus des informations déjà fournies :

- Sa page personnelle.
- Sa description.

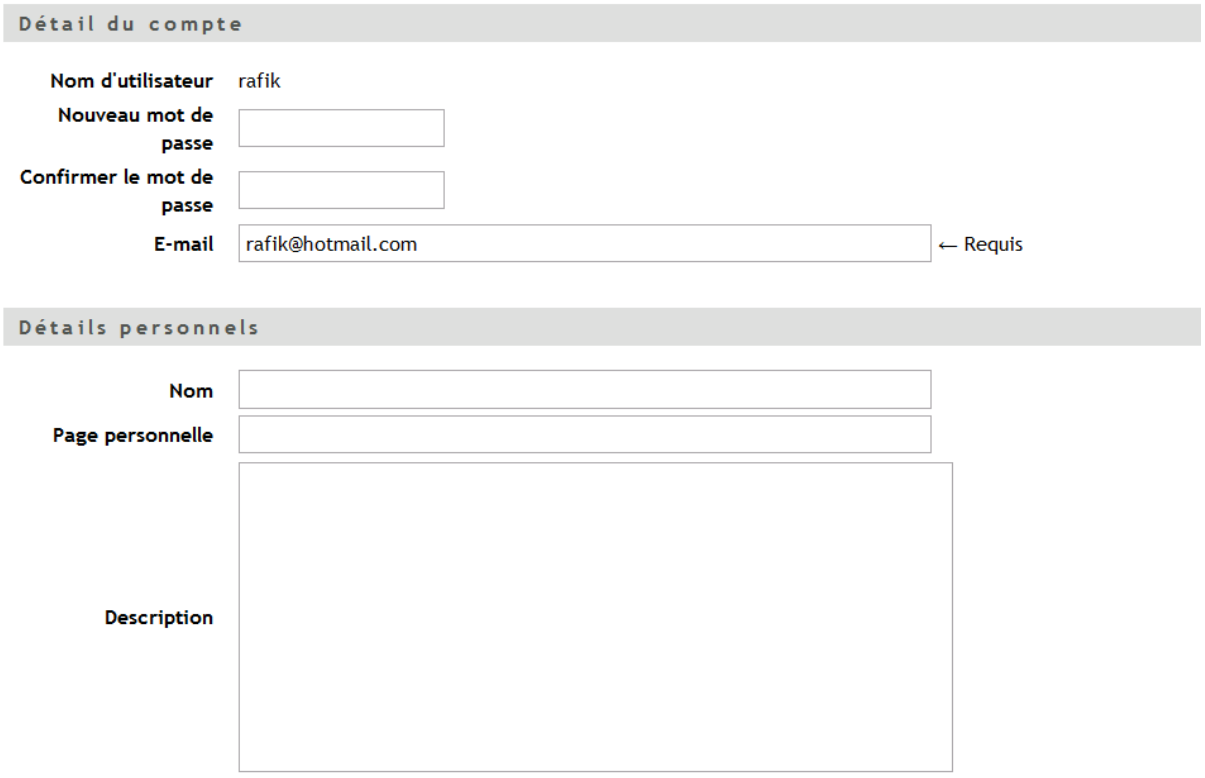


Figure IV.7. Page permettant l'édition de son profil

IV.4 Recherche d'un signet dans ses contacts selon ses préférences

L'utilisateur peut rechercher des signets partagés par des contacts.
Il lui suffit de :

- D'entrer le nom du signet ou une description.
- De sélectionner le contact ayant partagé le signet.
- De valider l'opération.

Figure IV.8. Mini Moteur de recherche permettant de rechercher des signets selon la préférence et le profil de l'utilisateur

IV.5 Ajouter des contacts

L’utilisateur peut aussi ajouter des contacts à sa liste en :
Sélectionnant le nom du contact.
Valider l’opération.

V. Conclusion

Dans ce chapitre une présentation de l’environnement et les outils utilisés pour implémenter cette entreprise a été effectué. Les principales interfaces de l’application ont été exposées et expliciter.

Conclusion générale :

Le Web sémantique constitue un axe de recherches émergeant. Diverses approches ont été proposées. Ces approches sont passées d'une recherche basée mots clés (correspondance syntaxique de la requête avec les descriptions des services Web) aux méthodes basées sémantique (degré de correspondance sémantique de la requête avec la sémantique des descriptions des services Web). Cependant, vu la diversité des utilisateurs et des conditions dans lesquelles ils accèdent aux services Web, d'autres paramètres doivent être considérés lors de la découverte, tel que le type du dispositif utilisé ,les préférences de l'utilisateur (ses centres d'intérêts) , sa localisation …etc. Tous ces paramètres forment un contexte d'utilisation particulier.

Dans ce travail nous avons proposé de décrire les différents paramètres du contexte de l'utilisateur et des services à travers les ontologies : du contexte de l'utilisateur, de service Web, de qualités de service et de paramètres de dispositif. Ensuite, autour de ces ontologies, nous avons conçu une architecture à base d'agents pour la découverte des services Web sémantiques basée sur le contexte.

A court terme, nous allons implémenter notre proposition d'architecture. Afin de valider notre travail, nous effectuerons des tests avec différents utilisateurs et un panel de services Web. A long terme, nous ferons des extensions à l'architecture pour permettre la composition contextuelle des services Web.

Cette annexe se réfère à la spécification XML schéma du W3C qui comprend trois parties :

- La première partie décrit des méthodes pour structurer et valider un document : XML schéma tome 1 : structures. [Sit, 04] ce sont les ″composants ″ pouvant être élaborés à partir des types de données et utilisés pour décrire la structure de l'élément, de l'attribut et de la validation d'un type de document.
- La seconde partie définit les types de données : XML Schéma tome 2 : **type de données.** [Sit, 05] Ce sont les ″atomes″ utilisés comme base pour tous les composants les plus importants d'un schéma.
- Une partie contenant des exemples pour illustrer les règles d'emploi de XML schéma vu la complexité du standard. [Sit, 03].

Il existe également deux types de balisage dans les schémas XML qui correspondent à ces deux parties de la spécification :

- Les définitions : qui créent de nouveaux types (simples ou complexes).
- Les déclarations : qui décrivent les modèles de contenu des éléments et des attributs dans les documents.

Bien que les structures aient été définies avant les types de données, nous commencerons par ces derniers car nous en ferons usage dans les exemples sur les structures.

I. les types de données primitifs

Comme déjà souligné, un des grands avantages des schémas est la prise en compte du typage. On distingue les types de données primitifs et dérivés.

I.1. les types de données primitifs

Ces types constituent la base des autres types. Ils sont utilisés pour les valeurs d'éléments ou d'attributs mais ils ne peuvent pas contenir des éléments enfants ou des attributs. Le tableau A.1 suivant recense quelques-uns de ces types, il en existe bien d'autres.

Type primitif	**Signification**	**Exemple**
string	Séquence de caractères	<élément>ceci est une chaîne</élément>
boolean	Indicateur à deux états	<élément flag=’’true’’/>
décimal	Nombre décimal en précision arbitraire	<élément num1= ’’-1.23’’ num2=’’56’’/>
timeDuration	Durée de temps	<élément durée=’’P12A10M2JD0H40M27S ’’/> Durée de 12 ans, 10 mois, 2 jours, 0heures, 40 minutes et 27 secondes, (P pour période et D pour durée).

Tab.A.1 : Types primitifs de XML Schéma.

1.2 Types de données dérivés

Un type de données *dérivé* est défini selon un type de données existant qu’on appelle type de base pouvant à son tour être primitif ou dérivé.

Exemple

A partir du type de string, qui est un type de base, on peut dériver les types suivants : type symbole, type instance et type classe.

- Le type symbole.

```
<xs:simpleType name="Symbol">
   <xs:restriction base="xs:string" />
</xs:simpleType>
```

- Le type instance.

```
<xs:simpleType name="Instance">
   <xs:restriction base="xs:string" />
</xs:simpleType>
```

- Le type classe

```
<xs:simpleType name="Class">
    <xs:restriction base="xs:string" />
 </xs:simpleType>
```

Il s'agit là d' exemples de *structure* de schéma XML : définition de type simple que nous allons aborder au paragraphe relatif aux structures.

1.3. Facettes des types de données

Tous les types de données de XML Schéma comprennent un *espace de valeur* dont chacune est désignée par un ou plusieurs caractères dans *l'espace lexical* qui est l'ensemble des représentations de ces valeurs, par exemple, la représentation lexicale d'un chiffre **2** est une chaîne de caractères mais sa valeur est le concept mathématique '' deux ''. Les facettes définissent les propriétés de l'espace des valeurs, ses valeurs individuelles ou les éléments lexicaux.

Une facette est un aspect de la définition d'une valeur simple. Les valeurs peuvent être restreintes par rapport à une liste de possibilités, à des limitations, à un modèle d'expression régulière, à une longueur ou à un comportement. On distingue deux types de facettes : *fondamentales* et de *contraintes*.

- **Les facettes fondamentales**

Ce sont les propriétés abstraites qui définissent les caractéristiques sémantiques de l'ensemble des valeurs d'un type de données. Ces facettes sont au nombre de cinq : *égalité, ordre, bornes, cardinalité, numérique/non numérique.*

- **Les facettes de contrainte**

Ces facettes limitent l'espace de valeur d'un type de données dérivé. Voyons ci-après les principales d'entres elles.

Pattern (modèle)

Un pattern est une expression régulière qui contraint l'espace lexical d'un type de données.

Exemple : date sous forme : '' 16-10-1978''

```
<simple Type name =''Date''>
<restriction base=''string''>
    <Pattern value=''\d{2}-{2}-d{4}''/>
</restriction>
</simple Type>
```

Enumération

Si une valeur n'est pas spécifiée dans l'énumération elle ne sera pas valide. Ici l'ordre dans lequel apparaissent les éléments dans le document XML n'est pas important.

Exemple

```
<xs:simpleType name="slot-type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Any" />
        <xs:enumeration value="Boolean" />
        <xs:enumeration value="Integer" />
        <xs:enumeration value="Float" />
        <xs:enumeration value="String" />
        <xs:enumeration value="Symbol" />
    </xs:restriction>

</xs:simpleType>
```

II. Les structures

Les structures sont les composants pouvant être élaborés à partir des types de données. Les déclarations permettent de décrire des types de données simples ou complexes (£ II.1), des modèles de contenu (£ II.2), des types d'éléments (£.II.3) et des attributs d'éléments (£.II.4).

Cette description sert à valider un document à l'aide d'un parseur validateur, prenant en charge les schémas XML.

II.1 Définition de types de données

On distingue les types de données *simple* et *complexe.*

- Définition de type simple

C'est un ensemble de contraintes sur l'espace de valeurs et sur l'espace lexical du type de données. Ces contraintes sont une restriction du type de base ou de la spécification d'un type *list* limité par une autre définition de type simple.
Pour cela, l'élément <simple Type> est utilisé, ses attributs sont décrits dans le tableau A.2 suivant :

Attributs	**Description**
name	Nom du type simple
base	Nom du type de données primitif
abstract	De type booléen, s'il est à vrai c'est que le type ne peut être la définition d'un type dans une déclaration d'élément.
derivedBy	A pour valeur list ou restriction, utilisé pour définir le type simpleType

Tab.A.2 : Attribut de <simpleType>

Exemple 1 : dérivation par restriction

```
<simpleType name=''EntierNegatif '' base= ''integer''>
    <maxInclusive value=''-1''/>
</simple Type>
```

Le nouveau type simple EntierNégatif est dérivé du type integer (la valeur par défaut de drivedBy étant restriction ce dernier peut être omis).

Exemple 2 : dérivation par liste

`<simpleType name=''ListeElementDecimaux '' base='' décimal'' deriveBy = '' list ''/>` cet exemple crée une liste illimitée de nombres décimaux. Un élément se conformant à ce schéma peut être :

```
<ListeElementDecimaux>1 1.20 40</ListElementDecimaux>
```

II.2 Définition de type complexe

Utilisé pour définir une structure de type qui contient un ensemble de déclaration d'éléments, de références à des éléments et de déclarations d'attributs. Pour cela, l'élément <complexType> est utilisé ses attributs sont décrits dans le tableau A.3 suivant :

Attributs	Description
name, base, abstract	Ont la même signification que pour le type simple
derivedBy	Sa valeur doit être extension (ajout d'éléments à un type complexe) ou restriction (on restreint le type existant sur le nombre d'occurrences, l'intervalle de valeurs possibles, etc.)
content	A comme valeurs : elementOnly, empty, mixed, textOnly
block	Indique si la déclaration d'élément d'un modèle de contenu doit ou non valider des éléments particuliers

Tab.A.3 : Attributs de <complexType>

Exemple :

```
<xs:complexType name="classe">
    <xs:sequence>
        <xs:element name="nom Classe" type="xs:string" />
        <xs:element name="nom Slot" type="xs:string" />
        <xs:element />
     </xs:sequence>
  </xs:complexType>
```

II.2. Modèles de contenu

Un modèle de contenu est la description formelle de la structure et du contenu admissible d'un élément.

Exemple le modèle de contenu d'un élément <nomSlot> est décrit comme suit :

```
<xs:complexType name="nomSlot">
    <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="ASSOCIATED-FACET"
            type="Instance" />
        <xs:element maxOccurs="-1" minOccurs="0" name="DIRECT- SUBSLOTS"
            type="Instance" />
        <xs:element maxOccurs="-1" minOccurs="0" name="DIRECT-SUPERSLOTS"
            type="Instance" />
        <xs:element maxOccurs="1" minOccurs="0" name="DIRECT-tyPE" type="Class" />
        <xs:element maxOccurs="-1" minOccurs="0" name="DOCUMENTATION"
            type="xs:string" />
        <xs:element maxOccurs="1" minOccurs="0" name="NAME" type="xs:string" />
```

```
        <xs:element maxOccurs="-1" minOccurs="0" name="SLOT-CONSTRAINTS"
            type="Instance" />
        <xs:element maxOccurs="-1" minOccurs="0" name="SLOT-DEFAULTS"
            type="xs:anyType" />
        <xs:element maxOccurs="1" minOccurs="0" name="SLOT-INVERSE"
            type="Instance" />
        <xs:element maxOccurs="1" minOccurs="0" name="SLOT-MAXIMUM-CARDINALITY"
            type="xs:integer" />
        <xs:element maxOccurs="1" minOccurs="0" name="SLOT-MINIMUM-CARDINALITY"
            type="xs:integer" />
        <xs:element maxOccurs="1" minOccurs="0" name="SLOT-NUMERIC-MAXIMUM"
            type="xs:float" />
        <xs:element maxOccurs="1" minOccurs="0" name="SLOT-NUMERIC-MINIMUM"
            type="xs:float" />
        <xs:element maxOccurs="-1" minOccurs="0" name="SLOT-VALUE-TYPE"
            type="xs:anyType" />
        <xs:element maxOccurs="-1" minOccurs="0" name="SLOT-VALUES"
            type="xs:anyType" />
      </xs:sequence>
    </xs:complexType>
```

Il existe un cas particulier qui est le type complexe dérivé de type simple. C'est le cas où l'on veut déclarer un élément ayant un attribut et contenant une valeur simple. Comme par exemple : <édition siteweb=''www.eyrolles </édition>, c'est un type simple auquel on rajoute un attribut. Cet élément est déclaré comme suit :

```
<element name=''édition''>
  <comlexetype>
   <simplecontent>
     <extension base=''string ''>
       <attribute name=''siteweb'' type=''string''/>
     <extension>
   </simplecontent>
  </complextype>
</element>
```

- **cardinalité : minocures et maxoccurs**

Déterminent le nombre minimum et maximum d'occurrences d'un élément.

- **groupes de choix, de séquences ou non ordonnés**

<choice> est un élément de groupement qui doit apparaître dans l'ordre spécifié.

<all> spécifie les enfants devant être présents, sans qu'il ait à apparaître dans un certain ordre.

II.3 déclarations d'éléments

Les déclarations d'éléments permettent de fournir une description utilisable pour la validation, d'offrir des contraintes de valeur et d'établir des relations de contrainte entre les éléments reliés et les attributs.

Un élément de type complexe peut avoir des sous éléments et être qualifié par des attributs, alors qu'un élément de type simple ne peut avoir ni sous éléments ni attributs (les attributs font l'objet du paragraphe suivant)

Les principaux attributs qu'on utilise avec <element> sont décrits dans le tableau A.4 suivant.

attribut	**description**
Name	Nom de l'élément déclaré
Ref	Nom d'un élément préalablement défini
Type	Tout type de données valides intégré ou dérivé par l'utilisateur
Minoccurs, maxoccurs	Nombre minimal (maximal) d'occurrences autorisées
Default	De type sting, defini la valeur par défaut d'un élément
Fixed	Défini la valeur constante (valeur par défaut de l'élément)
Id	De type ID, identifiant unique de l'élément
Nullable	Cet attribut prend la valeur true ou false (null ou notnull) selon que l'élément apparaît ou pas

tab.A.4 : Attributs de <element>

Exemple :

```
<element name=''scénario'' type=''typeScénario''>
< !—déclaration de son type qui est complexe→
   <xs:complexType name="ScénarioType">
      <xs:complexContent>
         <extension base="THING">
            <xs:sequence>
               <xs:element fixed="STANDARD-CLASS" maxOccurs="0" minOccurs="0"
                  name="METACLASS" type="xs:string" />
```

```
            <xs:element maxOccurs="unbounded" minOccurs="0" name="Elte-sc"
                type="Instance" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="Eltp"
                type="Instance" />
            <xs:element minOccurs="0" name="nom-sc" type="xs:string" />
            <xs:element minOccurs="0" name="num_sc" type="xs:integer" />
            <xs:element maxOccurs="unbounded" minOccurs="0"
                name="précéder_par" type="Instance" />
            <xs:element maxOccurs="0" minOccurs="0" ref="SEPARATOR" />
          </xs:sequence>
        </extension>
      </xs:complexContent>
    </xs:complexType>
</element>
```

II.4 déclarations d'attributs

Les déclarations d'attributs associent à un nom d'attribut un type de donnée simple et spécifique (un attribut ne peut être que de type simple). Ces déclarations permettent de fournir une description utilisable pour la validation, de limiter les valeurs d'attributs à un certain type et de demander ou d'empêcher sa présence.

L'élément <attribute> possède plusieurs attributs dont certains sont décrits dans le tableau A.5.

Attribut	**Description**
Name	Nom de l'attribut déclaré
Ref	Nom d'un attribut préalablement défini
Type	Tout type de données **simple**s valides, intégré ou dérivé par l'utilisateur
Use	Cet attribut indique comment et si les attributs doivent apparaître au sein des éléments décrits <element> qui constitue l'élément parent de l'attribut
Value	De type string, définit la valeur par défaut de l'attribut
Id	Identifiant unique de l'attribut

Tab.A.5 : Attributs de <attribute>.

Les attributs légaux de l'attribut use sont : default, fixed, optional, requiered et prohibited

default : attribut déclaré comme valeur par défaut (précisée par l'attribut value).

fixed *:* une valeur constante est attribuée à l'attribut (précisée par l'attribut value).

optional : l'attribut est facultatif, c'est la valeur par défaut de l'attribut use.

requiered : l'attribut est obligatoire et doit toujours être utilisé avec l'élément parent.

prohibited *:* aucun attribut ne doit se trouver dans l'élément parent.

Exemple : reprenons l'exemple précédant sur les scénarios et améliorons le schéma de telle sorte qu'un scénario ait un attribut IDEN le qualifiant.

```
<element name=''Scénario'' type=''typeScénario''>
< !—déclaration de son type qui est complexe→
   <xs:complexType name="ScénarioType">
      <xs:complexContent>
         <extension base="THING">
            <xs:sequence>
               <xs:element fixed="STANDARD-CLASS" maxOccurs="0" minOccurs="0"
                  name="METACLASS" type="xs:string" />
               <xs:element maxOccurs="unbounded" minOccurs="0" name="Elte-sc"
                  type="Instance" />
               <xs:element maxOccurs="unbounded" minOccurs="0" name="Eltp"
                  type="Instance" />
               <xs:element minOccurs="0" name="nom-sc" type="xs:string" />
               <xs:element minOccurs="0" name="num_sc" type="xs:integer" />
               <xs:element maxOccurs="unbounded" minOccurs="0"
                  name="précéder_par" type="Instance" />
               <xs:element maxOccurs="0" minOccurs="0" ref="SEPARATOR" />
              <attrbibute name= ″IDEN″ type=″string″ use=″required″/>
             </xs:sequence>
          </extension>
       </xs:complexContent>
    </xs:complexType>
</element>
```

Un document conforme à ce schéma pourrait être :

```
   <Scénario id="Se_00336">
      <Elte-sc>Instance(hat_00928_of_Cls(Eltc))</Elte-sc>
      <Elte-sc>Instance(hat_00927_of_Cls(Eltc))</Elte-sc>
      <Elte-sc>Instance(hat_00932_of_Cls(Eltc))</Elte-sc>
      <Elte-sc>Instance(hat_00935_of_Cls(Eltc))</Elte-sc>
      <Elte-sc>Instance(hat_00929_of_Cls(Eltc))</Elte-sc>
      <Elte-sc>Instance(hat_00931_of_Cls(Eltc))</Elte-sc>
      <Elte-sc>Instance(hat_00933_of_Cls(Eltc))</Elte-sc>
      <Elte-sc>Instance(hat_00934_of_Cls(Eltc))</Elte-sc>
      <Elte-sc>Instance(hat_00926_of_Cls(Eltc))</Elte-sc>
      <Elte-sc>Instance(hat_00930_of_Cls(Eltc))</Elte-sc>
      <Eltp>Instance(hat_00925_of_Cls(Eltc))</Eltp>
      <Eltp>Instance(hat_00780_of_Cls(Eltc))</Eltp>
```

```
        <Eltp>Instance(hat_00781_of_Cls(Eltc))</Eltp>
        <nom-sc>L'algèbre_relationnelle_et_langage_SQL</nom-sc>
        <num_sc>28</num_sc>
 <précéder_par>Instance(Se_00325_of_Cls(Scénario))</précéder_par>
    </Scénario>
```

III. associer des schémas à des documents XML

Un schéma XML est constitué d'un préambule suivi des différentes déclarations. Le préambule est un groupe d'attributs situé au sein de l'élément racine <schéma>. Les espaces de nom sont utilisés pour faire références à plusieurs vocabulaires.

Exemple

<xs :schema xmlns :xs=''http://www.w3.org/2001/xmlschéma''
xmlns :xsi=''http://www.w3.org/2001/xmlschema-instance''/>
< !--différentes déclarations-->
</xsd:schema>

Les deux attributs utilisent les espaces de nom XML pour identifier les deux schémas du W3C utilisé dans presque tous les schémas XML. Le premier inclut les types fondamentaux tels que <element>, <attribute>, <simple Type> et <complex Type>, le second définit les types de données standards tels que string, flot, etc.
Le premier espace de nom est celui utilisé par défaut et peut donc être omis : <schéma>...</schéma>.

Il existe des APIs (Application Programming Interface) supportant les schémas conformément aux spécifications du W3C. Ces APIs permettent de valider la conformité d'instances de documents XML par rapport à une grammaire décrite par un schéma, par exemple Xerces .

Pour ce faire, on doit spécifier au sein de l'élément racine du document XML (soit <scénaio>) la localisation du schéma associé (soit scénario.xsd), ceci en utilisant l'attribut xsi : schemalocation.
Voici la syntaxe de cette déclaration :
< ?xml version=''1.0'' ?>

<scénario xmlns :xsi=’’http://www.w3.org/2001/XMLShema-instance’’
xsi:noNamespaceSchemaLocation=’’scénario.xsd’’>
<!- - le contenu du document XML- ->
</scénario>

Dans ce cas aucun espace de nom cible n’a été déclaré (xsi : noNameSpaceShemaLocation) car l’espace de nom utilisé est celui par défaut.

IV. Contraintes d’intégrité

Certains éléments des schémas XML permettent d’exprimer des contraintes *d’’unicité* et *de référencement* à des clés.

IV.1 Unicité

L’élément <unique> permet de définir un élément, un attribut (précédé par ’’@’’) ou une combinaison des deux.

La différence avec une clé est que l’unicité est vérifiée quand les éléments ou attributs sont présents.

Exemple : la valeur de l’élément <nom-sc> doit être unique.

<unique name=’’unique1’’/>
<selector Xpath=’’scénario’’/>
<field Xpath=’’scénario/nom-sc’’/>
</unique>

La balise <selector> indique les éléments concernés par la définition, la balise <field> précise quant à elle l’élément ou l’attribut dont la valeur doit être unique

IV.2. Définition d’une clé

Une clé est définissable sur un élément, un attribut ou une combinaison des deux. La valeur de la clé doit être présente, ne doit pas avoir la valeur ’’nulle’’ et doit être unique.

Exemple

<element name=’’didacticiel’’>

```
.......
  <key    name=''index''>
     <selector   xpath=''scénario''/>
     <field        xpath=''@IDEN''/>
  </key>
```

La validation vérifie si chaque scénario possède un IDEN unique.

IV.3 Référence à une clé

Toute clé référencée doit être définie au préalable. Ces concepts de clé et de leurs référencements ressemblent aux clés primaires et étrangères des BDD SQL.

Exemple

```
 <element      name=''didacticiel''>
.......
< !- - définition de la clé- ->
<keyref   name=''keyref1''    refer=''index''>
    <selector   xpath=''enseigner par BDD''/>
    <field   xpath=''num-sc/@iden''/>
</keyref>
</element>
```

Un exemple de document XML respectant ce schéma peut être :

```
<didacticiel>
  <scénario  iden=''2-212-09248-2''>
        <nom-sc> Edition de lien </nom-sc>
        <eltp> Modes d'adressage </eltp>
  <scénario>
……….
</didactiel>
```

1. Présentation d'UML

Face à la diversité des méthodes d'analyse et de conception objet et, en particulier aux différentes notations des mêmes concepts, UML (Unified Modeling Language) représente un réel facteur de progrès par l'effort de normalisation réalisé.

En effet, UML constitue une étape importante dans la convergence des notations utilisées dans le domaine de l'analyse et la conception objet puisqu'elle représente une synthèse des trois méthodes OMT, BOOCH et OOSE. Ces trois méthodes couvrent environ la moitié du marché des méthodes objet.

Par ailleurs, l'adhésion à UML de grandes entreprises comme Microsoft, IBM ou encore ORACLE montre bien l'intérêt qui est porté à cette évolution dans le monde de l'objet.

Rappelons qu'UML a déjà son histoire qui peut être retracée (voir figure B.1) rapidement. C'est à la fin de l'année 1994 que James Rumbaugh et Grady Booch décident de travailler ensemble à l'élaboration d'une méthode unifiée d'analyse et de conception objet. En 1995, Ivar Jacobson les rejoint en apportant notamment le concept des cas d'utilisation.

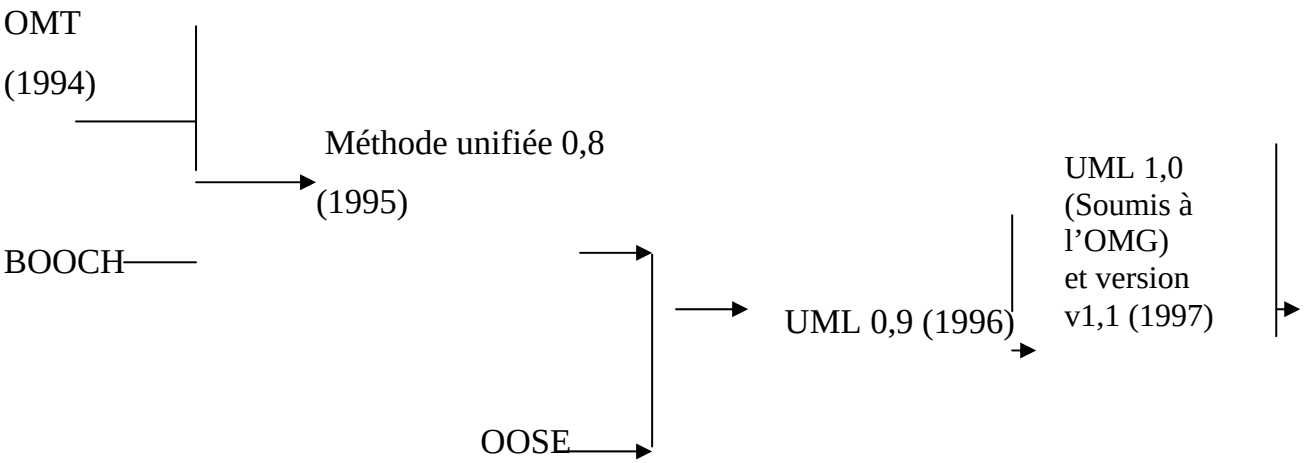


Figure .B.1: Les étapes d'élaboration d'UML

2. Les concepts de l'UML

Avant de décrire les diagrammes d'UML et la démarche d'utilisation de ces diagrammes, il est nécessaire de rappeler les concepts de base sur lesquels on s'appuie pour modéliser un système d'information, quel que soit le langage. Leur définition s'est affermit au fil du temps et a bénéficié des apports du courant de l'orienté objet..

A. le concept objet : Pour pouvoir qualifier d' « objet » un élément perçu, il faut qu'il satisfasse trois principes.

Le principe de distinction *:* l'élément peut être repéré et distingué d'autres Objets environnants.

Le principe de permanence **:** l'élément a une certaine durée de vie.

Le principe d'activité *:* quand on distingue un élément, on lui reconnaît un rôle dans le domaine que l'on étudie.

Pour représenter un objet nous avons trois possibilités (voir figureB.1)

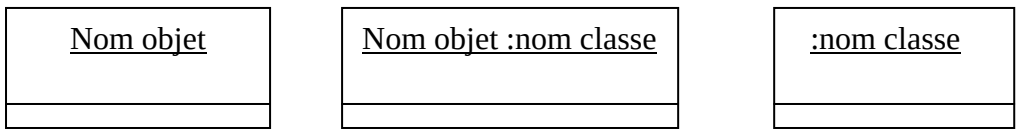


Figure.B.1: Les trois possibilités de représenter un objet.

B. le concept de classe : Le monde réel est constitué de très nombreux objets en interaction, les classes constituent des groupements d'éléments qui se ressemblent constituant ainsi des structures de haut niveau d'abstraction [Mul, 97]

Une classe est un type abstrait caractérisé par des propriétés (attributs et méthodes) communes à un ensemble d'objets et permettent de créer des objets ayant ces propriétés. Elle est représentée par un rectangle compartimenté comme le montre la figureB.2

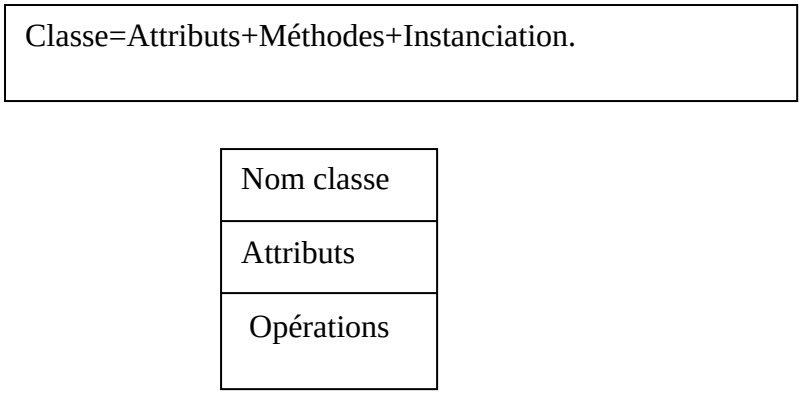


Figure.B.2: Représentation d'une classe en UML

C. Les relations entre les classes

A chaque famille de liens entre objets correspond une relation entre les classes de ces objets.

- **L'association**

Les associations représentent des relations structurelles entre classes, elles expriment une connexion sémantique bidirectionnelle entre les classes. En UML, cela est représenté par un trait plein entre deux classes [Lmp, 98]

On peut associer pour une association son nom, le rôle que joue chaque classe dans l'association ainsi que le nombre d'instances de la classe qui participe à l'association (cardinalité), Les valeurs de multiplicité conventionnelles sont : 0. 1, *, 0. *, 1..*. ??????

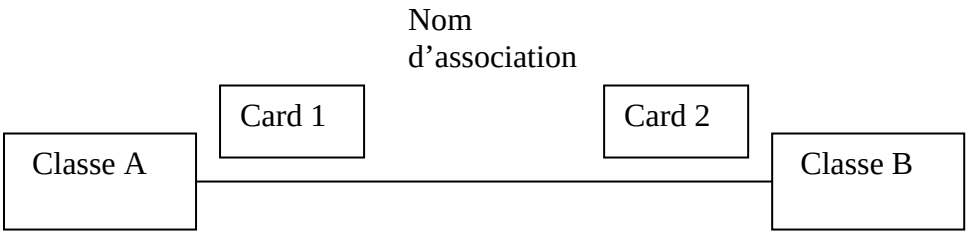
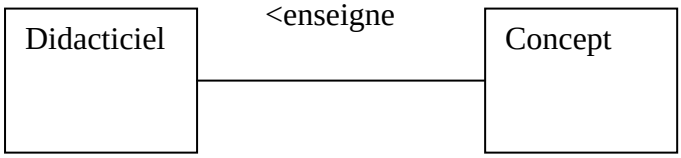


Figure.B.3: Représentation d'une association entre classes avec UML.

Exemple



Une association peut être d'arité supérieure à deux, dans ce cas, on la représente aux moyen d'un losange sur lequel arrivent les différents composants de l'association Comme l'illustre la figure.B.4.

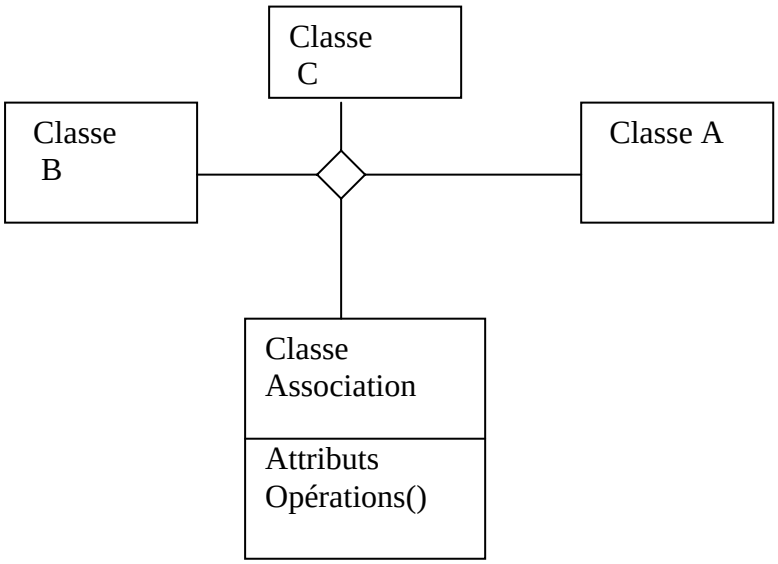


Figure.B.4: Représentation d'une association ternaire et d'une classe d'association

- **L'agrégation**

L'agrégation est une association où une des classes joue un rôle plus important que l'autre dans la relation, elle permet de représenter des relations de type tout et partie ou composé et composant (figure B.5)

L'agrégation se représente en ajoutant un losange du coté de l'agrégat. La composition (contenance physique) est représentée par un losange plein, dans ce cas la cardinalité du coté de l'agrégat est soit 0 soit 1 et la destruction du contenant entraîne celle du contenu.

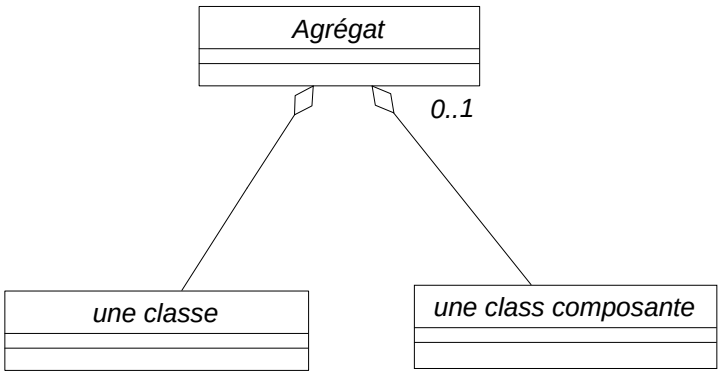


Figure.B.5: L'agrégation.

D. La hiérarchie de classes

- **La généralisation et la spécialisation :** La généralisation consiste à factoriser des éléments communs (attributs, opérations et contraintes) d'un ensemble de classes dans une classe plus générale appelé `super classe` [Mul, 97]
  La spécialisation est la fonction inverse de la généralisation, elle permet d'ajouter des propriétés spécifiques à une classe pour obtenir une sous classe plus spécialisée [Mul, 97] (figure.B.6)

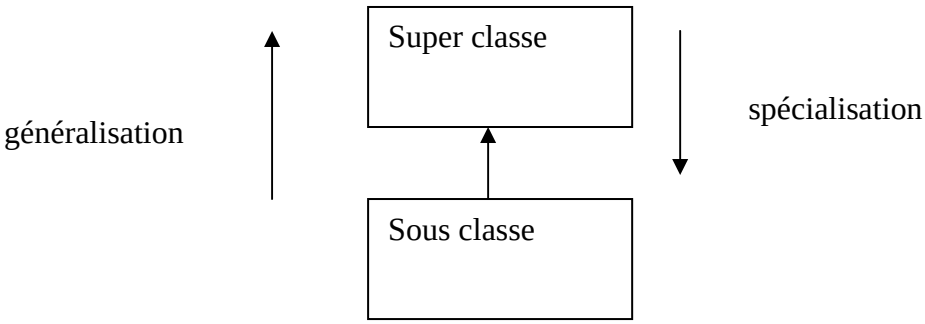


Figure.B.7: Représentation de la généralisation/ spécialisation.

En UML, on représente ceci par une flèche en forme de triangle pointant de la classe la plus spécialisée vers la classe la plus générale. On obtient ainsi, une classification des classes objets (voir Figure.B.8)

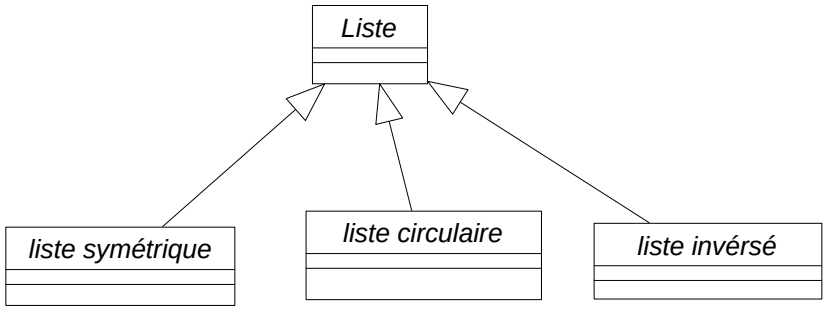


Figure.B.8: Exemple de classification de classes.

- **L’héritage**

L’héritage entre classes est une technique utilisée en programmation objet (POO) pour réaliser la classification. Ainsi l’héritage permet à une classe d’objets de réutiliser les attributs et les opérations définis pour une classe d’objets plus générale. On distingue : l’héritage simple et l’héritage multiple.

3. Les diagrammes d’UML

La modélisation consiste à créer une représentation incomplète mais exacte de la réalité : **Le modèle.**

Le méta modèle d’UML fournit une panoplie d’outils permettant de représenter l’ensemble des éléments du monde objet ainsi que les liens entre eux. Etant donné qu’une seule représentation est trop subjective, UML permet diverses projections d’une même représentation grâce aux vues.

Une vue est constituée d’un ou plusieurs diagrammes permettant de visualiser et de manipuler des éléments de modélisation. [Mul, 97]

on distingue deux types de vues illustrées par la figure.11 suivante [Site, 07] : La vue statique et la vue dynamique.

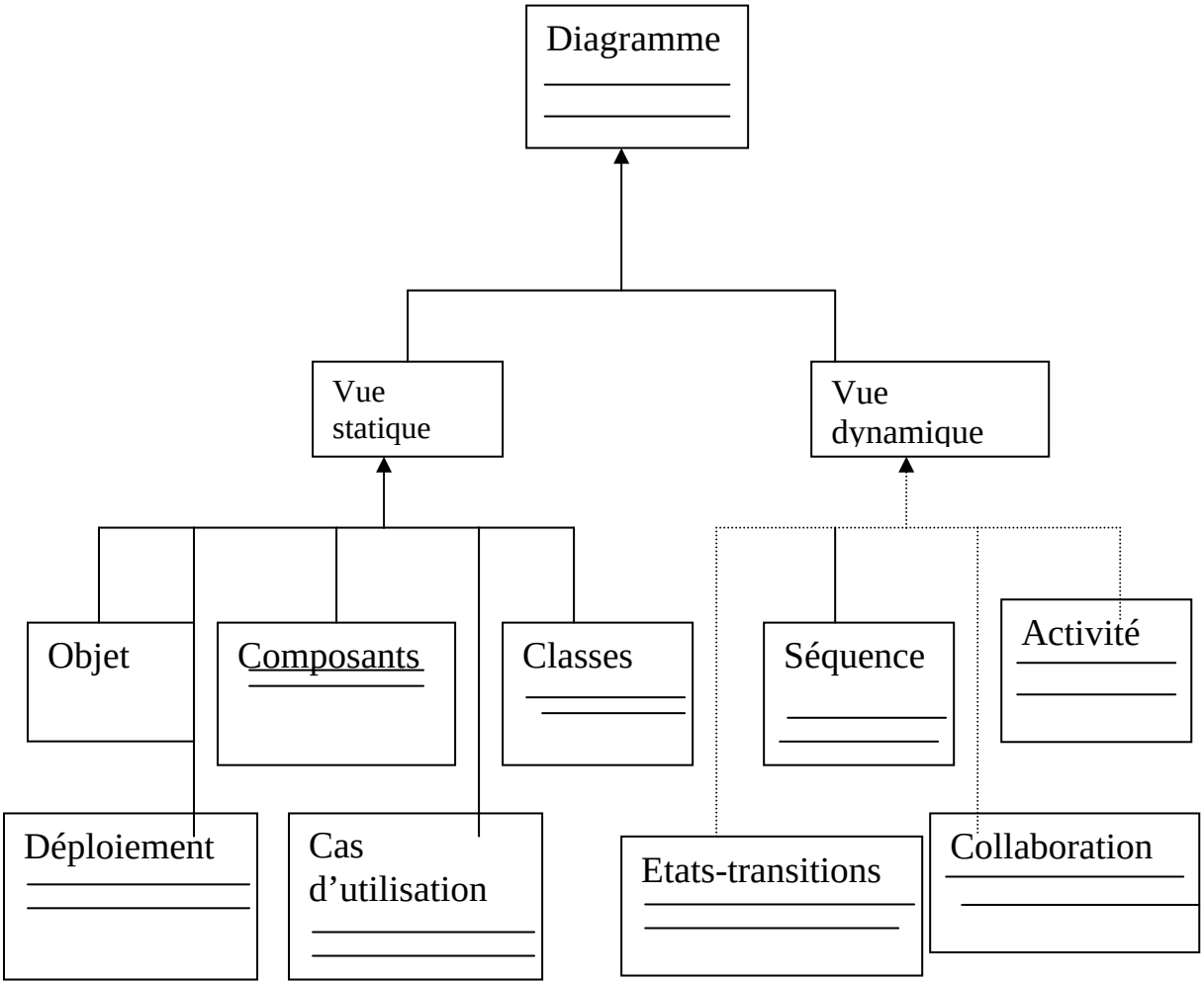


Figure B.9: Différents types de diagrammes définis par UML

Nous citons dans les paragraphes suivants les diagrammes que nous avons utilisés pour modéliser notre application.

3.1. Les diagrammes de cas d'utilisation (use cases)

Les cas d'utilisation décrivent le comportement du système, du point de vue utilisateur, Sous la forme d'actions et de réactions. Un cas d'utilisation indique une fonctionnalité du système déclenchée par un acteur externe au système [Site, 08]

- **Les cas d'utilisation**

Un cas d'utilisation (représenté par une ellipse en UML) se détermine en observant et en précisant, pour chaque acteur, les interactions et les scénarios de point de vue de

l'utilisateur. Un cas d'utilisation est vu comme une classe dont les instances sont les scénarios. Chaque fois qu'un acteur interagit avec le système, le cas d'utilisation instancie un scénario qui correspond au flot de messages échangés [Mul, 97] (voir figure.B.10)

- **Les acteurs**

Un acteur représente un rôle joué par une personne ou tout ce qui interagit avec le système, il est représenté par un petit personnage (voir figure.B.2)

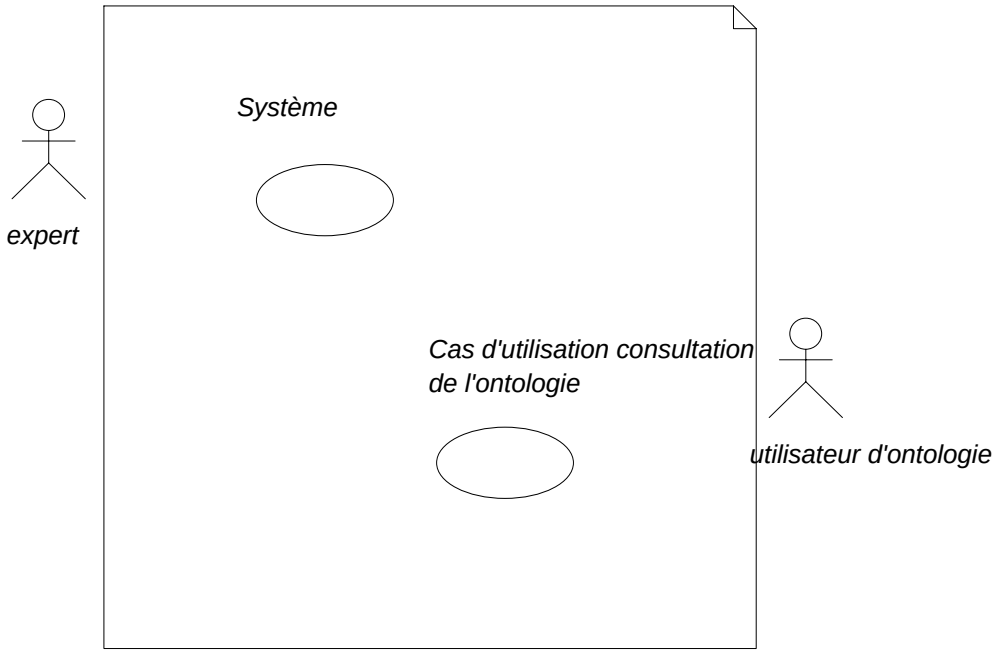
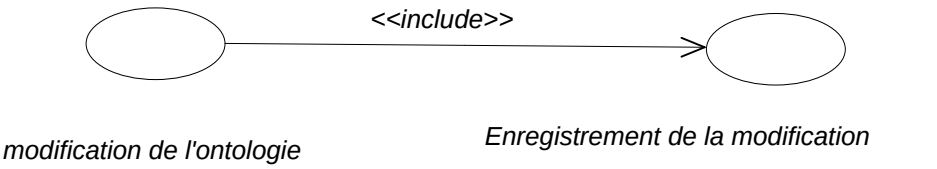


Figure B.10 : Diagramme de cas d'utilisation.

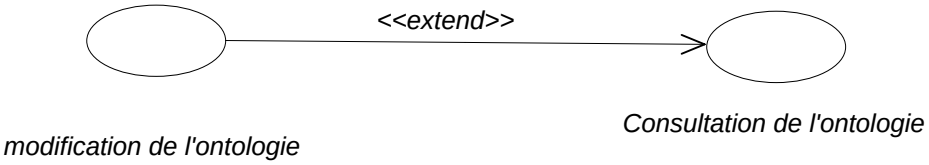
Les cas d'utilisation peuvent être en relation, trois types de relations sont définis en UML.

a. Relation de communication : entre un acteur et un cas d'utilisation.

b. Relation d'utilisation : une instance du cas d'utilisation source comprend également le comportement décrit par le cas d'utilisation destination



c. **Relation d'extension** : Le cas d'utilisation source étend le comportement du cas d'utilisation destination



3.2. Les diagrammes de classes

Les diagrammes de classes se représentent sous la forme d'un réseau de classes et d'associations, ce réseau modélise la structure d'un objet, son rôle au sein du système ainsi que ses relations avec les autres objets. [Lmp, 98]

Ces diagrammes sont la représentation de la structure statique en terme de classes et de relations.

3.3. Les diagrammes objets

Les diagrammes objets ou diagrammes d'instances montrent des objets et des liens entre ces objets. Ces diagrammes s'utilisent pour montrer un contexte particulier, par exemple avant ou après une interaction [Mul, 97]

Un objet est représenté par un rectangle qui contient le nom de l'objet souligné et éventuellement les valeurs des attributs dont le type est décrit au niveau de la classe (figure B.13)

Les liens entre objets qui sont instances de relations entre classes sont représentés par des traits munis de flèches si le sens de navigation est précise. La représentation d'une structure par des objets est plus parlante que celle abstraite représentée par des classes.

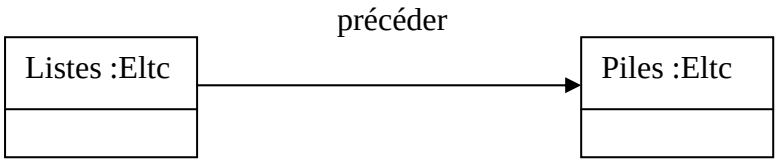


Figure B.11: Représentation des objets et des liens entre eux.

3.4 Les diagrammes de séquence

Ces diagrammes montrent des interactions entre objets selon un point de vue temporel.

On insiste ici sur la chronologie des envoies de messages. [Mul.97]

Les objets sont placés sur la première ligne en associant à chacun d'eux une barre verticale pointillée : ligne de vie. L'axe de temps est dirigé du haut vers le bas et les messages sont représentés par des flèches horizontales allant de l'émetteur vers le destinataire.

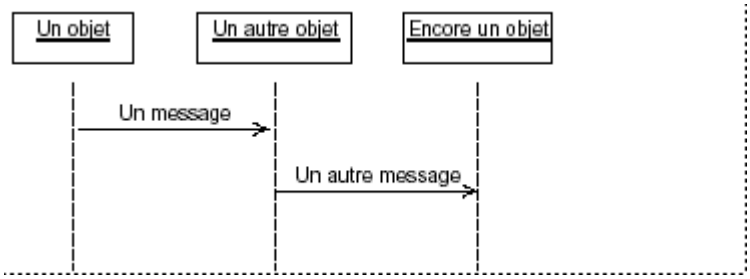


Figure B.12: Exemple de diagramme de séquence.

Le temps pendant lequel un objet effectue une action peu être représenté par des périodes d'activité sous forme de bandes rectangulaires. Dans ces diagrammes, on peut indiquer les branchements conditionnels par du pseudo-code placé le long de la ligne de vie ou entre crochets sur le message conditionné. On peut également placer en pseudo code les boucles d'interaction (while, for …)

4. Les outils de développement en cas d'UML

Il y a plusieurs outils qui supportent le formalisme UML aujourd'hui sur le marché : Certains sont de simples outils de dessin et d'autres des outils sophistiqués de modélisation objet.

4.1-Together : TOGETHER permet de cibler les plates formes suivantes : Java, C^{++}, IDL, Visual Basic 6, Visual Basic.NET.

4.1.1Together Control Center

Celui-ci profite d'une infrastructure commune et d'un environnement de développement intégré intuitif pour modéliser et construire des applications et aider les équipes à collaborer efficacement. Together Control Center regroupe la conception, le développement et le déploiement en une suite qui simplifie et intègre l'analyse, la conception, l'implémentation, le déploiement et le débogage d'applications.

4.1.2 Together Edition for JBuilder

Il étend les fonctionnalités de JBuilder Entreprise pour prendre en charge la modélisation, la construction et le déploiement de projets logiciels. Les modifications apportées dans JBuilder sont répercutées dans Borland Together et inversement.

4.1.3 Together Edition for WebSphere Studio

Il étend les fonctionnalités de IBM WebSphere Studio pour prendre en charge la modélisation, la construction et le déploiement de projets logiciels

4.1.4 Together Solo

Together Solo comprend les fonctionnalités élémentaires de développement d'applications de Together ControlCenter, comme l'éditeur de diagrammes UML, l'éditeur de programmation, GoF et d'autres patterns, ainsi qu'une ingénierie aller-retour simultanée pour Java, C++, Microsoft Visual Basic 6, Visual Basic .NET et IDL.

4.2 Rational Rose

Rational Rose est le Leader Mondial en outil de Modélisation UML, le Rose Link procure une liaison bidirectionnelle synchronisée entre le modèle UML de Rose et du code Java ou Delphi par exemple. Avec cette combinaison le reverse engineering

à partir d'une application Java ou Delphi est possible. Rose Link Java est disponible pour Borland, JBuilder, Visual Café, Oracle JDeveloper, & IBM's .

4.2.1 Rationnal Rose et Delphi : En optant pour UML, sans avoir encore choisi de langage ou d'environnement de Développement, En effet, Delphi est l'un des environnements de développement le plus utilisé au monde, et il est un outil idéal pour créer des applications performantes et robustes, grâce à son véritable Compilateur d'exécutables intégré. D'autre part Delphi utilisant un Langage Objet, c'est pile dans la lignée de la philosophie UML.

4.2.3.Rhapsody Modeler:

Un ensemble d'outils de modélisation UML qui permettent de générer du code objet de manière très paramétrable pour Java, C, C^{++} et ADA et de tracer l'exécution des modèles UML dont il est issu (Version US).

Références bibliographie :

[1] Philippe Laublet, Philippe Laublet, Jean Charlet, « *Introduction au Web sémantique* », Document électronique, PDF, Paris-Sud, Revue I3, 14p, http://leo.saclay.inria.fr/publifies/ gemo/GemoRepport-415.pdf.

[2] Michel Gagnon, « *Introduction au web sémantique* », École Polytechnique de Montréal, document électronique,2004,http://professeurs.polumtl.ca/michel.gagnon/…/tutorielSWIG04 pdf.

[3] Ta Tuan Ahn, « *web sémantique et réseaux sociaux Construction d'une mémoire collective par recommandation mutuelle et représentation* », thèse final de doctorat, ENST. INFRES Informatique et Réseaux, document électronique .PDF, p.24-29,2005,http:// pastel.enst.fr/1312.

[4] Bosak J., Bray T, « XML and the second-Generation Web », Scientific American, 280(5), 1999.

[5] T*R. Gruber*. « *Toward principles for the design of ontologies used for knowledge sharing*». Presented at the Padua workshop on Formal Ontology, March 1993.

[6] Mihoubi, H. « une Approche Déclarative De Traduction D'ontologies », thèse de doctorat, Université Stendhal-Grenoble ,2000.

[7] « *Terminologie Et Intelligence Artificielle* ». (Actes Du Colloque De Nantes, 10-11 Mai 1999) in Rint, réseau international de néologie et de terminologie, n° 19, 1999, document électronique PDF.

[8] Welty, C. « The Ontological Nature Of Subject Taxonomies ». In. Formal Ontology In Information Systems (N. Guarino Ed.).IOS press, Amsterdam, 1998.

[9] Frédéric Fùrst. Rapport de recherche « l'ingénierie des connaissances » institut de recherche en informatique .Nantes, Octobre 2002.

[10] Bachimont B., « *Modélisation Linguistique et Modélisation Logique Des Ontologies: L'apport de l'ontologie formelle* », Actes des Journées Francophones d'Ingénierie des Connaissances (IC'2001), Presses Universitaires.Grenoble, 2001.

[11] Uschold.M, King.M Moralee. S Zorgios.Y. «*The Entreprise Ontology*», 1995.

[12] Kassel G., Ontospee, « *une Méthode de Spécification Semi-Informelle d'Ontologies* », Actes des Journées Francophones d'Ingénierie des Connaissances (IC'2002), pages 75-87, 2002.

[13] Bernarers-Lee T., Hendler J.& Lassila O., « The Semantic Web», in Scientific American, Mai 2001.

[14] « *Terminologie et Intelligence Artificiell »,* (Actes Du Colloque De Strasbourg, 10-03 Mars 1999), réseau international de terminologie, n° 13, 1998.

[15] Phuc Hiep LUONG, « *Web sémantique* », thèse de doctorat, Ecole des mines. Spécialité : Informatique, temps réel, robotique, automatique, Paris, 2007.

[16] Yannick Prié, Serge Garlatti, « *Méta-données et annotations dans le Web* », Université Claude Bernard. Lyon, ENST Bretagne, document électronique PDF, 2004, http://liris.cnrs.fr/~yprie/ens/04-05/MasterRecherche/i3-AMD-WS.pdf.

[17] Jean-François Baget, Étienne Canaud, Jérôme Euzenat et Mohand Saïd-Hacid .INRIA, Montbonnot Saint-Martin, « *Les langages du Web sémantique* », CNRS, Université Claude Bernard. Lyon 1 .France .document électronique PDF, http:// citeseerx.**ist**.psu.edu/viewdoc/download?doi=10.**1**.**1**.80.2174.

[18] Lucie Philippe, « *les réseaux sociaux et les outils web* », Mémoire Master 2 professionnel, Université paris VIII. Spécialité : Gestion de l'information du document, Option : écriture hypertextuelles et publication en réseau. Saint-Denis, 2007-2008.

[19] Gaëlle Lortal, « un collecticiel pour une coopération via l'annotation de documents numériques », thèse de doctorat, Université des technologies. Troyes. Novembre 2006.

[20] Berrut , C.,Denos,N ,2003. « *Filtrage collaboratif* », in assistance intelligente à la RI,Hermes-Lavoisier,Chapter 8,pp30.

[21] Maltz, D., Ehrlich., 1995 . « *Pointing the Way: Active Collaborative Fitering* », Proceeding of CHI 95, p.7-11.

[**22]** Koivunen, M.R, Swick, Kahan, Prud'hommaux,. « *Annoteashared Bookmark* », KCAP workshop, Sanibel, Florida, 2003.

[23] Rucker,J.,J Polanco,M.J. « *Siteseer: personalized navigation for the web* », Communications of the ACM,vol.40,num 3,p.73-75, 1997.

[24] Guillaume Erétéo, Fabien Gandon, Michel Buffa, Patrick Grohan Relecteurs Talel Abdessalem, « *Analyse des réseaux sociaux et web sémantique: un état de l'art* ».

[25] P. Mika, « *Ontologies are us: A unified model of social networks and semantics, in The Semantic Web* ». Proceedings of the 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, volume 3729 of Lecture Notes in Computer Science, p. 522–536: Springer. November 6-10.

[26] F. Alkhateeb, J.F. Baget, J. Euzenat, « *RDF with Regular Expressions*». INRIA RR-6191, 2007, http://hal.inria.fr/inria-00144922/en.

[27] Olivier Corby, « *Graph Path in SPARQL* », INRIA. http://wwwsop.inria.fr/edelweiss/software/corese/v2_4_0/manual/next.php,

[28] J. Scott, « *Social network analysis* ». Deuxième edition, Edition Sage. (2000).

[29] L.C. Freeman, « *Centrality in social networks: Conceptual Clarification* », 1979.

[30] N. J., « *On Centrality in a graph* ». Scandinavian Journal of Psychology 15:322-336.

[31] P. Holme, B. J. Kim, C. N. Yoon et S. K. Han, « *Attack vulnerability of complex Networks* », Phys. Rev. E 65, 056109 2002.

[32] H. Kim, S. Yang, S. Song, J. G. Breslin and H. Kim, « *Tag Mediated Society with SCOTOntology* ». ISWC2007. 2007.

[33] Drifa Aaddour et R.Ahmed Ouamer, « *actes des rencontres sur la recherche en informatique* », Tizi-Ouzou 12-14 juin 2011 pp 98-105.

[34] Ahmed Ouamer et Hammache A, « *Un système de recherche d'information pour l'e-learning* », Revue Document numérique, Hermès Lavoisier,vol.11 numéro 1-2,(2008)85-105.